



IIT Madras

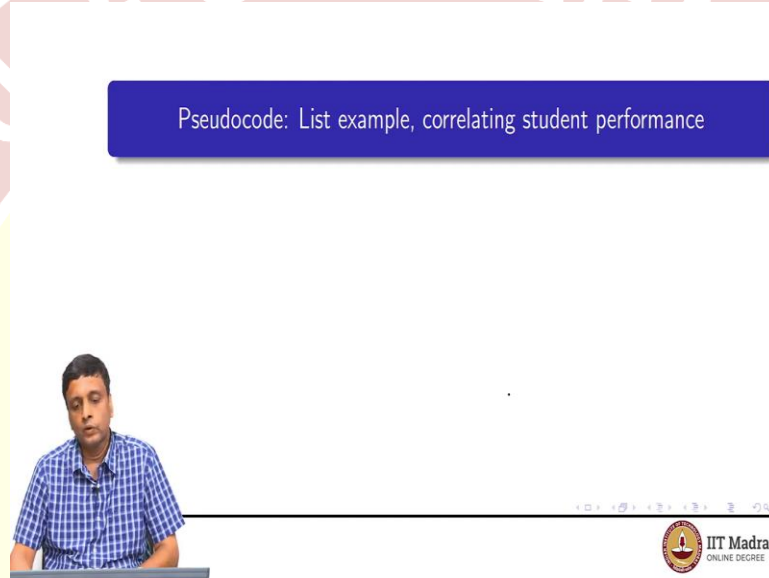
ONLINE DEGREE

Computational Thinking
Professor. Madhavan Mukund
Department of Computer Science
Chennai Mathematical Science

Professor. G Venkatesh
Indian Institute of Technology, Madras

**Pseudocode for systematic process of hypothesis verification to find relation between
Mathematics and Physics marks using lists**

(Refer Slide Time: 0:14)



So, we are seen how to use list use for which each to alliterate through a list we saw a simple example where we look at the topper across all the subjects. And then we also saw how to implement insertion sort which is a way of arranging the elements in the list either in ascending or in descending order. So, now let us put all this together in what substantial example. Which is to correlate the performance of students.

(Refer Slide Time: 0:41)

Correlating marks in Maths and Physics

- We want to test the following hypothesis
Student who perform well in Mathematics perform at least as well in Physics
- Assign grades {A,B,C,D} in both subjects
 - "Perform well in Maths" — grade B or above
 - "Perform at least as well in Physics" — Physics grade \geq Maths grade
- Algorithm
 - Assign grades in each subject
 - Construct lists of students with grades A and B in both subjects — four lists
 - Count students in A list for Maths who are also in A list for Physics
 - Count students in B list for Maths who are also in A list or B list for Physics
 - Use these counts to confirm or reject the hypothesis



Pseudocode: List example, correlating student performan

2/9



So, we are interested in this hypothesis that students who do well in Mathematics do at least well in Physics. So, we need a way of course of objectively stating this English sentence. So, one way is to say that students get, if they get a certain mark in Mathematics they get a higher mark in Physics. But we felt that marks are too fine grades somebody if gets 87 in Math's and 86 in Physics need not be considered to be worst in Physics and in Maths.

So, what we said was instead supposing we assign grades, so we assign grades A, B, C, D in both the subjects. Then what we would like is that those who perform well in Mathematics so say they get the top two grades A or B. Then they must get at least the same grade, so that same grade higher in Physics. So, if they get A in Maths they must get A in Physics because there is no higher grade.

But if they get a B in Maths they must get an A in Physics because there is no higher grade. But if they get a B in Maths then they must get a B or A in Physics. They can do as well which is to get a B or better it's to get an A. So, this is our question we want to ask whether this is true or not? Is it true that students who do well in Maths do at least well in Physics in terms of this quantification over grades?

So, this is the algorithm that we proposed first we have to of course assign grades because we have only marks. So, our grade data, the score data for students has only marks so how do we assign grades? Having assign the grades, then we need to come up with students who have A's

and B's? So, in particular we need all those students who got A in Maths, those who got a B in Maths, those A in Physics, those who got B in Physics. So, there are 4 list of students that we have to considered once we decide how to award grades.

Now we have to do this hypothesis validation. We have to check whether this hypothesis is true or not. So, the first step is to check all those students who have gotten A in Maths in order for the hypothesis to be true those who have gotten A in Maths should also gotten A in Physics. So, you want to count how many students who have in A in Maths also have in A in Physics. So, this gives you a number of students from that category who support the hypothesis.

Secondly, you go to the list of B grades in Maths and now you check how many of these student have either a B or an A in Physics. So, these two counts will give us a list of how many students confirm the hypothesis. Those who are not in this list but who have gotten A or a B in Maths will be those who do not confirm the hypothesis.

And therefore, by counting how many of them confirm versus how many of them do not confirm. We can come up with some measure of how valid the hypothesis is, if an overwhelming number of confirm, then we can say the hypothesis is true such a hypothesis is never going to be 100 percent true. Because this is based on human beings and human beings have different kinds of this which drive them. So, you cannot expect that such a hypothesis will have 100 percent confirmation.

But say for example 80 percent of the students follow this, then you can say that it is reasonable if only 50 percent do it than it is like a random chance only half of them get better in so the other half get worse. So, there is really no correlation between Maths and Physics so that is what, so we are not presupposing we want to find out from the data whether this hypothesis is valid or not for which we need to execute this computation.

(Refer Slide Time: 4:08)

Assigning grades

- Assign grades {A,B,C,D} approximately at quartile boundaries
 - Top 25% get A, next 25% get B, next 25% get C, bottom 25% get D
- To calculate quartiles, extract marks as a list and sort the list
- Need to identify the students — each entry in the marks list is a pair [StudentId,Marks]
- Procedure to extract marks information as a list for a subject

Get the marks lists for Maths and physics

```
Procedure BuildMarksList(field)
marksList = []
while (Table 1 has more rows) {
  Read the first row X in Table 1
  marksList = marksList ++ [[X.SeqNo, X.field]]
  Move X to Table 2
}
return(marksList)
End BuildMarksList
```

mathsList = BuildMarksList(Mathematics)
physicsList = BuildMarksList(Physics)

Pseudocode: List example, correlating student performance 3/9

IIT Madras
ONLINE DEGREE

So, the first step is in this computation as we said is to convert marks into grades. So, we are going to take a simple minded approach which is we have just going to give equal number roughly of A's, B's, C's, and D's. So, we are not going to look at the relative performance in terms of how well they have done compare to each other. But we are just going to ruthlessly cut them off at one fourth.

So, in our grades dataset we actually have 30 students, so roughly this means every group will have 7 or 8 students because you cannot divide 30. There were 32 then they would be 8 in each category since there are 30. Some will have 7 some will have 8. So, we first we need to get this quartile thing. So, to calculate these quartiles we need to find out who are the top one fourth then the next one fourth and so on.

So, the easiest way to do this as we said when we discuss sorting is to actually arrange this marks in ascending orders or in descending order. Once they are arrange in ascending order we can take the first group. The first one fourth in this list and call that the highest or lowest depending on the ascending or not. And the next highest in the next quartile and so on, so we can break up the sorted list into 4 equal segments and each segment will correspond to one grade.

So, one of the things that we will need is not just the list of marks because we need to check the marks in Physics against the marks in Mathematics. So, we need to know a student who has got in A in Physics an A in Mathematics whether he has gotten in Physics he or she has got a B in

Mathematics whether she has got a B in Physics or a A in Physics. So, we need to associate with each mark also the identity of the student. So, we have in the data set we have the sequence number of the card 0 to 29. So, we use that to identify the student. So, if says student number 7 has gotten A in Mathematics, we want to check if the same student number 7 is also gotten A in Physics.

So, that means if the list that we construct of marks which are going to sort does not have just marks as his entries it has this pairs, this pairs which we will write as a two element list. It has the first element of list is the student ID the sequence number and the second is the marks. So, now when we are comparing for getting the sorted marks we do not really care. The student ID is use only when we confirm the hypothesis, when we check every student who has an A whether they have got A there or not. But to actually award the A we are not looking at the student ID.

So, we will come back to this later on, but let first look at how to get this marks list for a given subject. So, it is easy enough we just who are usual thing we start with an empty list. And then we process the table and for every row in the table we just pull out two field the sequence number and we are now using this across subjects. So, we will use it for Mathematics, for Physics, for chemistry separately.

So, depending on which subject we provide, we can even do it for total marks if you want but we are not using it in this example. We are only using Mathematics and Physics but any field which is there on that card we can extract a list of the sequence number and that field value. So, this builds the marks list. So, what we want to do is initially build the marks list for Maths and the marks list for Physics.

So, will the marks list for Maths we will get sequence of pairs of this form so we will have S1 the sequence number of the first student in the marks of the first student. Then S2, M2, and so on. So, this is what is list will look like so this is no particular order this is just in order in which we process the table. Now we need to sort it.

(Refer Slide Time: 7:43)

Assigning grades ...


- Use insertion sort for `mathList` and `physicsList`?
 - Entries are `[id,marks]`
 - To compare `[i1,m1]` and `[i2,m2]`, only look at `m1, m2`
- Extracting values at the beginning and end of a list
 - `first(l)` and `last(l)`
`first([1,2,3,4])` is 1,
`last([1,2,3,4])` is 4
 - The remainder of the list is given by `rest(l)` and `init(l)`, respectively
`rest([1,2,3,4])` is `[2,3,4]`,
`init([1,2,3,4])` is `[1,2,3]`

Procedure SortedListInsert(l,x)

```
newList = []
inserted = False
foreach z in l {
  if (not(inserted)) {
    if (last(x) < last(z)) {
      newList = newList ++ [x]
      inserted = True
    }
  }
  newList = newList ++ [z]
}
if (not(inserted)) {
  newList = newList ++ [x]
}
return(newList)
End SortedListInsert
```

Modify SortedListInsert

Pseudocode: List example, correlating student performan 4/9



So, we could use insertion sort but as we said before the problem with insertion sort as we have defined it is that it does not have a natural way of comparing the entries in our list. Because now we do not just have marks we have this each entry is a pair ID comma marks. So, we want to look at `i1, m1` one pair `i2, m2` another pair and compare it but all we really interested in this whether `m1` is smaller than `m2` or not. We are not really interested in `i1` and `i2`.

So we need to be able to take this list and extract out only the `m1, m2` part. So, for this is useful to just add a new function or an operation which allow us to do this. So in general we might want to extract things at various parts of the list. But I think it is reasonable to assume that we can look at the first element and the last element of the list quite easily. So, let us call this `first` and `last`.

So, `first of l` will give us the first element in the list as it, as you would expect. And `last of l` will give us the last element of the list. So, if I have a list `1, 2, 3, 4` then `first of l` will be `1` `last of l` will be `4`. So, `first` give us the first item `last` give us the last item. So, if I take for example this kind of with a thing and I apply `first` to it I will get the ID. And if I apply `last` to it I will get the marks. Now it will be useful to also look at what happens to the `rest`.

So, if I do `first` and `last`, then I have the remaining part, so I take out the first then I have the second onwards so I will call that the `rest`. So, the `rest` is a list, `first` is an item, `rest` is the list. So, if I take `rest of 1, 2, 3, 4`, it is the list `2, 3, 4`. `First of 1, 2, 3, 4` is the item one it is not the list one it is the item one but the `rest` is a list. Similarly, if I take `last of 1, 2, 3, 4` I get `4` so I can take the

initial part which I will call init. So, in it of the list will be the list 1, 2, 3 everything expects the last one.

So, now that we have this in place, let us do this minor modification of our insert procedure which takes into account the special nature of our information. So, we do not just have simple numbers to compare but we have this pairs of numbers. So, what we do we have to change only one thing which is when we decide on whether to shift or not, whether we decide to insert x into the new list or by pass and go to the next z instead of comparing x to z directly.

So, remember now x will be of the form x will be of the form ix, mx . It will be some pair and this will be of the form iz, mz . So, what we want to do is compare mx and mz so this is why we use last. We say last of x smaller than last of z this is the only change we have to make. So, in general, if I apply sorting across different types of values the main thing I need to change is the comparison.

How do I compare this values? So, depending upon how by compare them that if condition alone changes everything else is the same. Once I have compared them everything else the same I shift them to the new list. I insert insertion sort everything is the same. So, this is the only change we need to make our original insertion sort in order to accommodate this special data that we have.

(Refer Slide Time: 11:04)

Assigning grades ...

```
■ InsertionSort uses updated  
SortedListInsert
```

```
sortedMathsList =  
    InsertionSort(mathsList)  
sortedPhysicsList =  
    InsertionSort(physicsList)
```

$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
D	C	B	A

Pseudocode: List example, correlating student performan 5/9

IIT Madras
ONLINE DEGREE

Once we have done this now we have built this Maths marks and the Physics marks by calling that build marks list procedure. So, then we can use this insertion sort the usual insertion sort but

with the new insert function that insert operation is updated to you is last instead of just x. so insertion using this updated sorted list insert will give us a sorted Maths list and the sorted Physics list.

So, once we have the sorted Maths list and the sorted Physics list we can get down to this business that we really interested in which is to assign grades. So, we are trying to assign grades so the reason we are sorting all this is because we want to take the sorted list. And we want to divided into 4 groups and say that this is A because this an ascending order, so the left most will be a D and the right most highest marks will be an A. And we want this to be roughly be one fourth. So, this is our strategy for awarding grades.

(Refer Slide Time: 11:59)

Assigning grades ...

- Assign grades to a sorted list by quartile
- Compute quartile boundaries based on class size

```
Procedure SimpleGradeAssignment()  
  classSize = length(l)  
  q4 = classSize/4  
  q3 = classSize/2  
  q2 = 3*classSize/4
```

Hand-drawn diagram showing a number line with values 75, 15, 22.5, 94, 63, 42. Above the line is a bracket labeled '26'. Below the line are values 94, 63, 42.

Pseudocode: List example, correlating student performan 6/9

IIT Madras
ONLINE DEGREE

Assigning grades ...

- Assign grades to a sorted list by quartile
 - Compute quartile boundaries based on class size
 - Initialize list for each grade
 - Assign grades based on the position in the list
- `SimpleGradeAssignment` returns a list containing four lists, for the four grades

```
Procedure SimpleGradeAssignment(l)
  q4 = ..., q3 = ..., q2 = ...
  gradeA = [], ..., gradeD = []
  position = 0
  foreach x in l {
    if (position > q2) {
      gradeA = gradeA ++ [first(x)]
    }
    if (position > q3 and position <= q2) {
      gradeB = gradeB ++ [first(x)]
    }
    if (position > q4 and position <= q3) {
      gradeC = gradeC ++ [first(x)]
    }
    if (position <= q4) {
      gradeD = gradeD ++ [first(x)]
    }
    position = position + 1
  }
  return([gradeA, gradeB, gradeC, gradeD])
End SimpleGradeAssignment
```



Pseudocode: List example, correlating student performan

6/9



So, we are going to create this simple grade assignment so I will tell you why it is a simple grade assignment in the minute but we are going to assign grades to a sorted list by this quartile. So, first we have to so quartile is just a position now that is sorted so quartile is just define by the position the first one fourth of the list is bottom most quartile. The next one fourth is the second is the third quartile. Next one fourth is the second quartile and so on.

So, the first thing to do is to identify these quartile boundaries. So, there is a function here which we have not seen before but which is obvious. So, we will assume with that with list we have this length function. So, the length function just tells us the number of element is in the list so that means we do not need to know an advance how many elements are there of course we can compute length by saying for each x and l and we can take a counter and say count equal to count plus one and we can find out the length.

So, is not the that we do not how to find the length just using what we already know but it is convenient to assume that we have this length function given to us. So, the class size is given by this length of this marks list say everybody is being accounted for so this l could be either the Maths list or the Physics list all we assume is that it is a sorted list. So, we assume that is sorted and in ascending order.

So, since it is sorted and it is a ascending order the first quarter is the fourth quarter. So, the boundary of the first quarter q4 is the class size divided by 4. So, supposing there are 30, then this divided by 4 this will give you 7.5 which is not an issue because as we go along, then the

boundary of the third quarter where the third quarter ends the second quarter begins is going to be 15 and the second and first quarter are going to be at three fourth of this. So, it is going to be a 22.5 so this is going to be q_4 this is going to be q_3 this is going to be q_2 so this give me positions with respect to the overall list where these greater boundaries come so that is the first thing I compute before I start assigning grades.

So, let us assume that we have done that now the next thing is that I have to construct this four list. Who all have got A, who all have got B, who all have got C, who all have got D? So, I will have four list grade A, grade B, grade C, grade D and I will assign them all to be empty. Now, I will go through the list and I will keep track of the position. So, I start with the zeroth position so it is conventional in computing to start with the zeroth position. So, I will start with the zeroth position.

And I will check whether that current position that I am looking that whether it is in the first quartile that is bigger than q_2 it is in the second quartile means less than or equal to q_2 and bigger than q_3 , third quartile is less than equal to q_3 bigger than q_4 or is in the fourth quartile is lesser equal to q_4 . So, this is very similar to what we did when we did this third max's. When we did this third max's, we tried to find out which zone the new value came in. So, here we are not going to update any thing we are going to insert it into the correct list.

So, if it is in the first quartile, then we put it into the grade A list. If it is in the second quartile we put in the grade B list. So, all these things are identical expect which list we are updating. So, the new x that we are processing does it go into grade A, grade B, grade C or grade D depends on whether its position is above q_2 , between q_2 and q_3 , between q_3 and q_4 or below q_4 . So in this way for each position I will do this I will keep incrementing the position. So, I will walk through this list from beginning to the end.

So, this an important factor about list that we have to keep in mind that when we do a for each basically it walks from beginning to end. There is no way to directly go to this seventeenth position. You have to walk through 16 positions to get to the seventeenth position. That is an unfortunate fact about list so list are the sequence which are built up in this sequence and you have read them in that sequence, for each will only allow you to go from beginning to end there is no way to directly jump it. That will be a different data structure which we see later on.

So, at the end of this so I have got this 4 grades grade A, grade B, grade C, grade D they are roughly equal. Now I have to give them back I have to tell you what the grades are. So, the way I will do it just to create a new list which consist of all this 4 list together. So, the simple grade assignment will return 4 list. The grade A list, the B list the C list and the D list. So, I will get back not one list but 4 list back together in a single list, a list of list. So, a list can contain anything particular can contain list so that is what this guys.

Now why is this a simple grade assignment well we already saw when we actually did with our data set when we did this by hand that this is not a very nice way to give grades because they could be marks at their boundaries which are duplicate. So, the person who the highest a C grade and the lowest B grade may have the same marks. We are not saying that all marks if all marks are distinct everybody gets a different mark, then at least you will have some consistency that two students with the same mark will get the same grade because they are no two students will same.

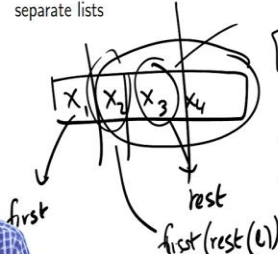
But if I have two 62 and one 62 happens to be at the top of the q_3 boundary and the other one is the bottom of the q_2 boundary one will get a B and one will get a C that is very unfair. So, what we did when did it by hand was we adjusted these boundaries by hand to go to nearest reasonable point where everybody with a so either we moved it one side so that the D grades became low or we move to the other side so the C grades became high, so depending on which way it look more reasonable we did some hand adjustment.

So, coding that hand adjustment is not very easy as you can imagine that is why we have assume this very simplistic thing but this is not really how you will assign grades even if you are going by quartile because you have to pay attention to this fact that students are the boundary of a quartile could have identical marks on either side. But anyway leaving that a side let us assume that this works. So, we have now taken the sorted list of Maths marks and created 4 list of Maths marks with A grade, B grade, C grade, D grades. Similarly, we have taken the physics marks.

(Refer Slide Time: 17:47)

Assigning grades ...

- Assign grades corresponding to Maths and Physics marks
- Unpack the four lists into four separate lists



```
mathsGrades = [A, B, C, D]
SimpleGradeAssignment(sortedMathsList)

physicsGrades = SimpleGradeAssignment(sortedPhysicsList)

mathsAGrades = first(mathsGrades)
mathsBGrades = first(rest(mathsGrades))
mathsCGrades = last(init(mathsGrades))
mathsDGrades = last(mathsGrades)

physicsAGrades = first(physicsGrades)
physicsBGrades = first(rest(physicsGrades))
physicsCGrades = last(init(physicsGrades))
physicsDGrades = last(physicsGrades)
```

Pseudocode: List example, correlating student performan 7/9

IIT Madras ONLINE DEGREE

So, we can take the Maths grades they can take the simple grade assignment pass it the sorted Maths list and it comes back with these four things. Similarly, we take the sorted Physics list comes back these 4 things. Now we have to unpack this four things. Because we do not want four list we want the A list, the B list, the C list, and D list separately because we need to iterate over them separately.

So, how do we unpack so this is where we use all those fancy things that we did. So, we talk about first and last and we also talk about init and rest. So, supposing I take a list, so supposing it has x_1, x_2, x_3, x_4 it has four elements. So, if I take first, then I get x_1 and then we said that this is the rest. But the rest is also list so if I take the first of the rest then I get x_2 . So, first of the rest is the second element in l.

So, what we are saying is that the Maths A grade, so remember the grades were given as A grades, B grades, C grades, D grades are list of four values so the first value in that list is the A grade so I take the first of the Maths grade is a, so this is got A, B, C, D it is got 4 list coming back. So, first of those list is the A list. Now, if I take the rest, that is everything except the first and I take the first of that then like we saw here that this the second value so that is the B grades. So, by using the first and first of rest I get the first value in the second value.

Similarly, from the other side if I take the last value I get the D grades and if I take the init of that then I get up to the third position and then the init, the last of the init is the third position. So, the

in it will give me 1, 2, 3 and last of that will give me 3. So, that will give me the C grades. So, by this clever use of first, and rest, and last, and init we can extract these 4 quantities.

Of course, we want the third you can get from the front also by telling rest and then taking rest of that. So, you take rest twice that knocks off two elements and then you take the first. So, by doing this combination of first and rest you can actually get to which ever position you want but in particular if we just want the first and the last is very easy. If we want the second last and the first second also is quite easy as you can see.

So, we do this for Maths we do this for Physics also so again the A grades and the Physics are the first the D grades are the last. The B grades are the first of the rest and the C grades are the last of the init. Now we are where we want to be this is what we our goal was we wanted to separate out the students into those who have got A in Maths, B in Maths, C in Maths, D in Maths and those who have got A in Physics, B in Physics, C in physics, and D in Physics.

So, after all this hard work by first extracting the marks from the data set into list then sorting the list, and then after sorting the list partitioning them into these quartiles and assigning grades. And then unpacking these grades back into 4 list we have finally got the list that we want. Now in this list of course we are not interested at all in the C and D grades because we only looking at the well performing student. So, students have got at least A or B but in the process of doing those four list we have got four more list. So, it is no harm anyway we will continue.

(Refer Slide Time: 21:08)

Test the hypothesis

- Check how many students with A in Maths confirm the hypothesis
 - `exitloop` prematurely terminates a `foreach` loop
- Check how many students with B in Maths confirm the hypothesis
- Finally check `length(confirm)` against `length(confirm)+length(reject)` to decide if the hypothesis holds

```
foreach x in mathsBGrades {
  found = False
  foreach y in physicsAGrades {
    if (x == y) {
      confirm = confirm ++ [x]
      found = True, exitloop
    }
  }
  if (not(found)) {
    foreach y in physicsBGrades {
      if (x == y) {
        confirm = confirm ++ [x]
        found = True, exitloop
      }
    }
  }
  if (not(found)) {
    reject = reject ++ [x]
  }
}
```



So, now we have got these 4 things. So, the first thing that we have to check is how many students who have got A in Maths have also got A in Physics. This is just our familiar foreach we have to do foreach x in Maths, foreach y in Physics check if x is equal to y only thing is we have to keep track whether we found the x or not. Earlier we were only keeping track one way we are only keeping track of those things that we found for in sense when we are looking for students who have born in May and live in Chennai, only if they satisfied both we added them into a list if they did not satisfy which through it way without remembering.

Similarly, when we are looking at students who have toppers in all the subjects if somebody was not at a topper in all the subject we did not care. But here we want to know how many confirm and how many do not confirm. So, for that reason we will actually keep track whether we found it or not we have a variable call found. So, if x is found in if x from Maths an A grade student in Maths is found in Physics, then we set found equal to true.

And for this same x if we exit this loop we go through all the Physics loop, a physics students and we have not found it if it is not found than we in, put it in a different list. So, we have a confirm list, confirm list are those students who have confirm our hypothesis reject list is those students who have rejected our hypothesis.

So, in this way using this variable found we can keep track of whether they are going the confirm list which is easy or whether we have not confirm them which is what this found is telling us. In which case we put them explicitly in the reject list, now one other advantage of doing this is that when we have found it, we can actually also stop looking at the rest. We are not going to find this student again.

We have not found it we have to continue but we have found the student in the Physic list obviously the student is not going to appear again so we can leave the loop. So, it is convenient to actually add a way to get out of this iteration. So, we will just assume that we have this extra command called exit loop which kind of abort this for each loop at this point and comes out. So, what does it do it goes to the foreach where you found it and it comes out to the end of that foreach so that is what I does.

So, the is also an outer for each so that foreach continues the exit loop is inside a foreach so you go to the nearest the for each inside with exitloop is and that for each it has thou foreach

abruptly terminated without processing the remaining elements. So, whatever you seen so far is done but it is like you skip ahead to the end and proceed so thou you have finished. Even though you did not see the remaining elements.

So, this is what happens for the A grades. So, for the A grades you check if they are in the A grades of Physics we have to do the same thing for the B grades except that we have to check for the B grades in Maths whether they are A in Physics or B in Physics. So the first part is the same so you will check so we have every x who has got a every student who is got a B in Maths. So, the Maths B grade we first check for every y in the Physics A grades.

So, as the student got as much as in A in Physics if so you are fine, again you set a variable found. Now you have not found in the A grades then you check in the Physics B grades again the same loop only you do it only if you have not already found it because it is wasteful to do this otherwise in both cases if you find it you exit. So, you exit the Physics A loop, you exit the Physics B loop, and finally if we have gone through both these loops are not found it either in those things you have variable found is still false. In that case you put it in the reject side.

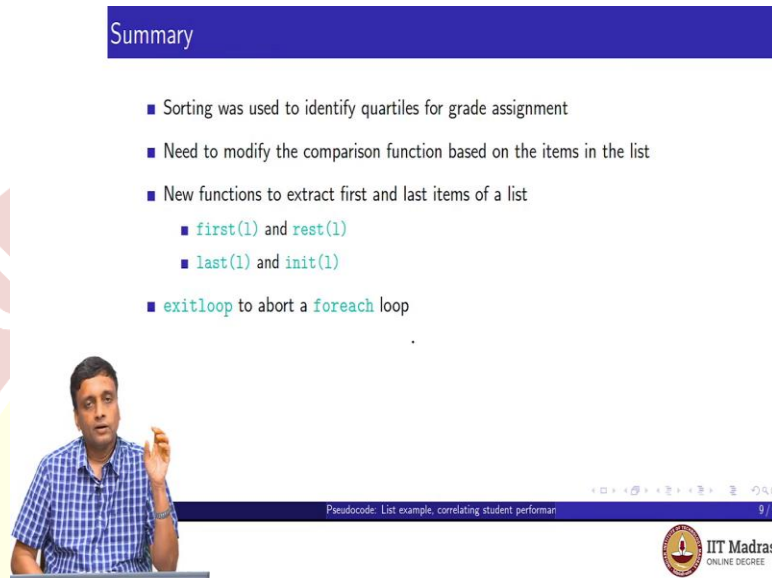
So, it is just two nested iterations one after the other. First check A, B with A and then check B with B. and if it is not in A and it is not in B then you put it in reject if you found it in the A or in B then you put it into confirm. So, at the end of this two loops, so you have two loops which are in kind of over wrote. So, the first loop said all those students who have got A in Maths whether they are confirm or rejected. Then all the students who has got B in Maths whether they are confirm or reject.

And after these two things now we have this two list confirm all those students who confirm the hypothesis all those students who reject to the hypothesis. Or what we want is a number we do not need to know who these students are, so as we said before we can take the length of that list. So, the length of the list confirm will tell us how many students actually had done at least as well in Physics as they are done in Maths and the length of reject will tell us those who did not.

So, this is the total number of students. So, if we check the length of the confirm list versus the total number of students will get a fraction. So, if the fraction is something like half it means that this is the totally random occurrence student who do Maths well in Maths may or may not do well in Physics. We have no correlation but if it is a large number like 0.8 or 0.9 it means that

there is a good correlation between Maths and Physics. So, that is the purpose of this whole exercise but among the way we have seen a lot of interesting things with list.

(Refer Slide Time: 26:04)



Summary

- Sorting was used to identify quartiles for grade assignment
- Need to modify the comparison function based on the items in the list
- New functions to extract first and last items of a list
 - `first(l)` and `rest(l)`
 - `last(l)` and `init(l)`
- `exitloop` to abort a `foreach` loop

Pseudocode: List example, correlating student performan 9 / 9

IIT Madras
ONLINE DEGREE

So, to summarize, we saw one concrete use of sorting here which is to identify these quartiles for grade assignment. So, if we did not do this then it will be very hard to figure out who are the top 25 percent who are the next 25 percent once we sort them we can just chunk that list into equal size bites of one fourth and we get the 4 grades, but one thing we saw is it because we needed to keep the list in a more complicated form at the elements are not single numbers not just the marks but these pairs of ID and marks, we need it to change our insert function to compare the second element of that and for this use this new notation of first and last and init and rest so that we can extract out selectively the beginning and the end of list and the rest of the list which is left after removing these extreme elements.

And finally, we saw that there is this exit loop which we can use when we want to explicitly abort for each loop because we have found what we wanted to find and there is nothing useful to be done by processing the lesser loop and one more thing which was not explicitly mentioned anywhere but which is used simplicity is this length function. So, we also use this length function which allows us to extract the length of a list without going through it.

Now remember this length function you can easily do, so you can say count equal to 0 and then say for each x in l count equal to count plus 1. So, at the end of this for every time you walk

through the list every element you saw you incremented count. At the end of this count is going to be the length. So, it is not length of 1 is anything very complicated to calculate but it is new sense to have to write this loop every time you want the length of list. So, is convenient to able to call this function.

