



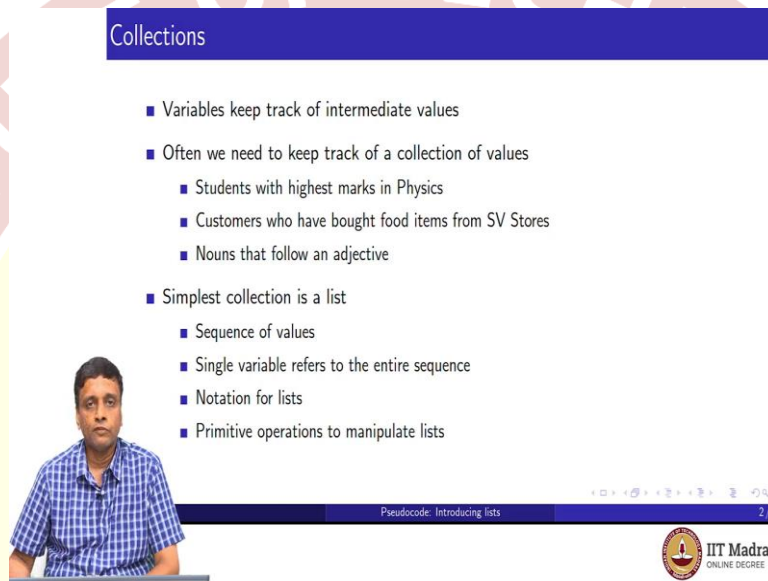
IIT Madras

ONLINE DEGREE

Computational Thinking
Professor Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute
Professor G Venkatesh
Indian Institute of Technology, Madras
Pseudocode for lists

So, we saw that we need to use lists to keep track of information. So, what we will do now is try to integrate these lists into the pseudocode that we have been writing for our procedures.

(Refer Slide Time: 0:26)



Collections

- Variables keep track of intermediate values
- Often we need to keep track of a collection of values
 - Students with highest marks in Physics
 - Customers who have bought food items from SV Stores
 - Nouns that follow an adjective
- Simplest collection is a list
 - Sequence of values
 - Single variable refers to the entire sequence
 - Notation for lists
 - Primitive operations to manipulate lists

Pseudocode: Introducing lists 2 / 5

IIT Madras
ONLINE DEGREE

So, we are dealing with collections of values, so as we saw when we started writing our pseudocode, we use variables to keep track of intermediate values in our computation to keep track of count, sum, and various other things that we need as we go along. But very often these intermediate values are not single values but collections of values. So for instance, if we are looking at students and their marks, there may be more than one student who has the highest marks in physics. So, we need to keep a collection of students in that category.

Or if we are doing customer behaviour, then the set of customers who have bought food items from say SV Stores is now a collection of names, it is not one single name. Or in our words dataset, we might be interested in knowing which nouns come immediately after an adjective. So, in all these cases the variables we are trying to keep track of are not keeping track of the single item, but a list or a sequence of items.

So, list is just a sequence of values and the advantage of using a list is we do not have to worry about how many values will actually be there in that sequence, whether it is 1 or 10 or 20, there is a single name that refers to this entire sequence and then of course once we have

that we need some notation to manipulate these sequences. So, what we will be exploring starting with this lecture is how to extend our pseudocode to give us effective ways of manipulating lists within the kind of syntax that we have been using for our computational procedures.

(Refer Slide Time: 1:51)

Pseudocode for lists

- Sequence within square brackets
 - [1,13,2]
 - ["Vedanayagam", "cane", "Monday", "school"]
 - [] — empty list
- Append two lists, $l_1 ++ l_2$
 - l_1 is [1,13], l_2 is [2,17,1]
 - $l_1 ++ l_2$ is [1,13,2,17,1]
- Extend l with item x
 - $l = l ++ [x]$
- Examples
 - List of students born in May

```

mayList = []
while (Table 1 has more rows) {
  Read the first row X in Table 1
  if (X.MonthOfBirth == "May") {
    mayList = mayList ++ [X.Seqno]
  }
  Move X to Table 2 ←
}
  
```

Pseudocode: Introducing lists 3/5

IIT Madras ONLINE DEGREE

So, the first question is how do you write a list? Well, that is natural enough, we use this square bracket notation to denote the beginning and the end of the list and we write the elements of the list between the square brackets using commas. So, very often our elements will be of the same type for instance, we have a list in the first example of integers 1, 13 and 2. In the second we have a list with 4 strings, Vedanayagam, cane, Monday, and school.

Now, it will not always be this way and we will not be very insistent on this, but most natural situations, a list will have an underlying base type that is, each element of a list will have a fixed type. An important list just like for numbers, 0 is an important value because we need to initialise it. Similarly, for a list, an important value is the empty list. The list that we start with which has no elements in it and this we will just write by a matching pair of brackets, square brackets which indicate that there is no content but it is a list.

So, now the first thing that we need to do is to be able to manipulate these lists. So, when we saw that when we had numbers and things like that, we had expressions, arithmetic operations to add numbers, we had ways of extracting things from numbers, so we need to do the same thing with list. So, the most basic thing you can do with a list is to put two list together to form a larger list. So, this we will call append and we will write it using this double plus notation.

So, if we have a list l1 and we have a list l2 and then we want to kind of stick them together one after the another, then we write that is l1 plus l2. So, for example if l1 is 1 and 13, it has two elements and l2 has 2, 17 and 1. So, remember that in a sequence, the order is important. So, this is not a set, this is a sequence. And in the sequence things can be repeated. So, we have 1 and 13 in l1, we have another 1 in l2. If we put it together we get 1, 13 followed by 2, 17 and 1 is a single 5 element list starting and ending with 1. So, this is how we combine list together.

Now, this is the general case but one very specific version of this that we will use is to add an element to a list. So, the most basic thing just like incrementing a number by 1, if you have a counter and you want to say count equal to count plus 1 or if you want to take a sum and you want to say sum equal to sum plus some quantity. The most basic thing you can do with a list is to add one more element at the end of the list. So, the way we will do that is use this plus plus so if you want and add a element x remember that plus plus requires two lists, it requires a left list and a right list, an element on its own is not a list, but if we put it within square brackets, it becomes a list of size 1.

So, this is a list of length 1. So, we take l plus plus square bracket x and this basically gives us a new list with one more element in it namely x at the last position. So, plus plus always remember, adds to the right. So, as an example, here is a simple piece of pseudocode which uses this notation in order to extract the list of students from our school database, we want the list of students who are born in May.

So, what we do is we initialise this list to be empty. So, you want a May list, May list is the list of all students born in May. So, we initially say there are no students born in May. So, this is like initialising a counter to 0. And now we use this Table format, so we assume that our data is given to us as a table with each row representing one student. So, while the table has more rows, we pick up the first row and then we check whether the month in that particular item that we have read corresponds to May or not.

So, we do not really have a field called month of birth in our data, we have date of birth, but let us assume that we can extract the month with this notation. So, we extract the month of birth and we check if it is May and if it is May, this is what we said before we want to extend our May list with this new guy, so we have an existing May list and we add to it the sequence number of the correct card. So, in this way we are accumulating the sequence numbers or the IDs of all the students who are born in May into this table, this list, May list.

And of course, as we read a row, we move it into another table so we do not have to process it again. And in this way we keep going until we exhaust the rows. So, this way we read all the rows in the table and from that we extract the relevant information, in this case the sequence number provided the column with month, date of birth has month May.

(Refer Slide Time: 6:22)

Pseudocode for lists

- Sequence within square brackets
 - [1,13,2]
 - ["Vedanayagam", "cane", "Monday", "school"]
 - [] — empty list
- Append two lists, l1 ++ l2
 - l1 is [1,13], l2 is [2,17,1]
 - l1++l2 is [1,13,2,17,1]
- Extend l with item x
 - l = l ++ [x]
- Examples
 - List of students born in May
 - List of students from Chennai

```

chennaiList = []
while (Table 1 has more rows) {
  Read the first row X in Table 1
  if (X.TownCity == "Chennai") {
    chennaiList = chennaiList ++ [X.Seqno]
  }
  Move X to Table 2 .
}

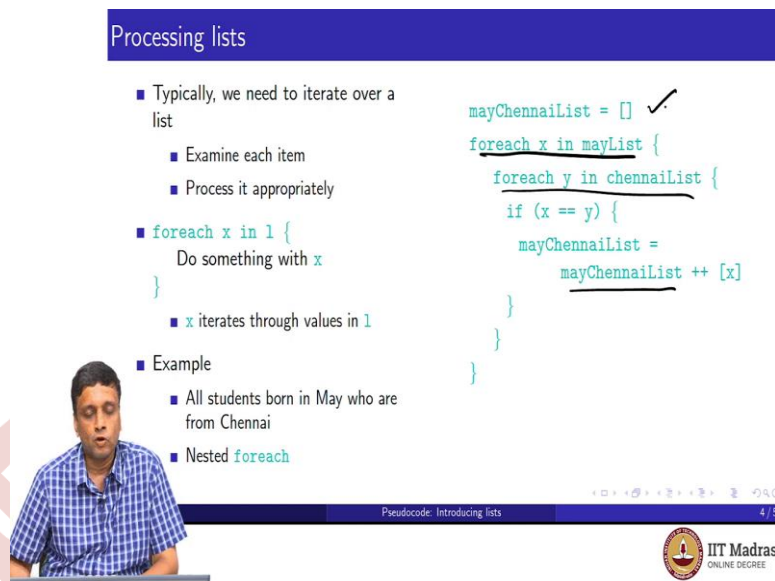
```

Pseudocode: Introducing lists 3/5

IIT Madras ONLINE DEGREE

So, you can do a similar thing for instance, with an other field. For instance, if you want to know which all students come from Chennai, then you again process the entire table from beginning to end and this time we will use a different list, let us call it Chennai list. So, initialise Chennai list to be empty. And then, as we go along, whenever we see a card in which the town or city is Chennai, then we append that sequence number to Chennai list. So, exactly the same as the one we did for date of birth in May.

(Refer Slide Time: 6:51)



Processing lists

- Typically, we need to iterate over a list
 - Examine each item
 - Process it appropriately
- `foreach x in l {`
Do something with `x`
`}`
 - `x` iterates through values in `l`
- Example
 - All students born in May who are from Chennai
 - Nested `foreach`

```
mayChennaiList = []  
foreach x in mayList {  
    foreach y in chennaiList {  
        if (x == y) {  
            mayChennaiList =  
                mayChennaiList ++ [x]  
        }  
    }  
}
```

Pseudocode: Introducing lists 4/5 IIT Madras ONLINE DEGREE

So, having got a list, what do we do with it? So, most often we need to go through the items in a list and process them in some way. So, we need to go through it systematically from beginning to end, look at each item and decide what to do with it. So, this kind of list iteration requires a primitive.

So, we are going to create one called `foreach`. So, `foreach` is a special kind of primitive which allows us to iterate over a list. So, when we say `foreach x in l`, it means that `x` will initially take the first value in `l`, then it will take the second value in `l`, then it will take the third value in `l`, so with each execution of the loop, `x` moves forward one position in `l`.

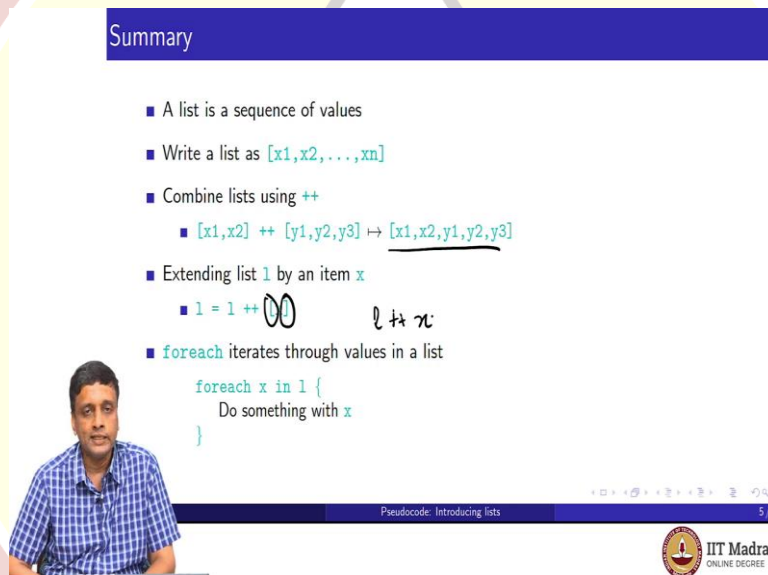
Of course, if `l` is empty, then this loop will not execute at all, it will terminate exactly like if you had a `while` loop in which the condition is initially false. So, if we have an empty list, this `foreach` loop will just exit otherwise, it will go through and it will pick up each element in `x` inside the list we will of course, do something with that `x`. So, in this way this `x`, the variable `x` that we assign in this `foreach` will iterate through the values in `l`.

So, as an example, we have constructed these two lists, the list of students sequence numbers who were born in May and the sequence number of the students who come from Chennai. So, if we wanted to find out those who qualify in both lists, those who are both born in May and are from Chennai, then we will use a familiar nested loop which is that for every sequence number which appears in the first list, we have to check whether it appears in the second list. So, this can be done using this new `foreach` operator just by nesting it.

So, we take every x which is there in the list of students who are born in May and then for each of those, we pick up all the y 's corresponding to the students who are from Chennai and if we find that the sequence number x matches a sequence number y from the second list, then we will append it to this new list we are building up which is called `mayChennaiList` that is people who are born in May and come from Chennai and we had of course, initialised this to empty, so to begin with we had no students who were both born in May and from Chennai.

And systematically, we did not go back, notice, we did not go back to the actual table. We already had these lists, May list and Chennai list and now by just doing a nested iteration using this `foreach`, we are able to extract the common elements. So, this is a typical way in which we process lists.

(Refer Slide Time: 9:19)



Summary

- A list is a sequence of values
- Write a list as $[x_1, x_2, \dots, x_n]$
- Combine lists using `++`
 - $[x_1, x_2] ++ [y_1, y_2, y_3] \rightarrow [x_1, x_2, y_1, y_2, y_3]$
- Extending list l by an item x
 - $l = l ++ [x]$
- `foreach` iterates through values in a list


```
foreach x in l {
    Do something with x
}
```

Pseudocode: Introducing lists 5/5

IIT Madras
ONLINE DEGREE

So, to summarise, a list is a sequence of values which we write within square brackets separated by commas. And we combine lists using plus plus. So, x_1, x_2 , a list with two elements plus plus y_1, y_2, y_3 , a list with 3 elements will give us a new list which has 5 elements in the same sequence left to right. So, the second list, second argument of plus plus is added to the right of the first argument and you basically, dissolve the boundary and make it a single list.

And as we saw one important use of this plus plus is to attach one new item to a list where we say l plus plus the new item within square brackets and these square brackets are important because without the square brackets if I write l plus plus x , this is not a well formed expression because the l plus plus x has a list on one side and it has a single item on the other side. So, we need two lists for plus plus, so that is why we have to put the square bracket.

And in order to process lists, we saw this foreach operation. S, foreach iterates through the values in list and this allows us to process lists systematically from beginning to end and do something with each element as we go along.

