

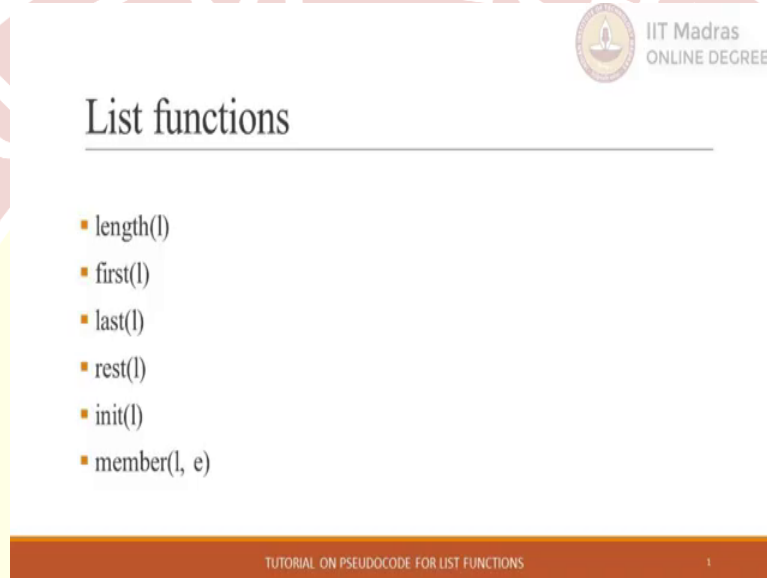


IIT Madras

ONLINE DEGREE

Computational Thinking
Professor. Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute
Professor. G.Venkatesh
Indian Institute of Technology, Madras
Tutorial on pseudocode for list functions

(Refer Slide Time: 00:25)



Hello Computational Thinking students. In this tutorial we will see pseudocodes of list functions used by professor in the previous lecture. This is the list of all those special functions; `length`, `first`, `last`, `rest`, `init` and we will also introduce a new function called `member`. Now we will see the computational logic involved in each of these functions. It is possible to write function definitions in many different ways. As we have learned earlier that the procedures with same interface can have different implementations.

(Refer Slide Time: 01:06)

length(l)



e.g. `length([20, 30, 40, 50, 10])` is 5

```
length(l) {  
    count = 0  
    foreach x in l {  
        count = count + 1  
    }  
    return(count)  
}
```

```
foreach x in Pile 1 {  
    }  
foreach x in Table 1 {  
    }
```

TUTORIAL ON PSEUDOCODE FOR LIST FUNCTIONS

2

`length`; is a function which counts the number of elements in a list. Therefore, it accepts a list as a parameter and returns an integer which represents the count of elements. Here we will iterate over the list `l` using this new operator called `foreach`. It will go through the entire list element by element and increment the count variable. If we pass an empty list as a parameter then it will return 0 because `foreach` block will not execute at all. If we observe carefully the functionality of this `foreach` operator is similar to the iterator logic which we use over a pile of cards or rows of table.

So far we have used `foreach` operator to iterate over the list and `while` loop to iterate over cards or table rows. But `foreach` operator can also be used instead of `while` loop like `foreach x in Pile 1` or `foreach x in Table 1` where `x` holds a card or a table row.

(Refer Slide Time: 02:26)

first(l)



e.g. `first([20, 30, 40, 50, 10])` is 20

```
first(l) {  
    foreach x in l {  
        return(x)  
    }  
}
```

TUTORIAL ON PSEUDOCODE FOR LIST FUNCTIONS

3

First function accepts a list as a parameter and returns the first element in the provided list. The return statement will terminate the execution of the foreach block while accessing the first element itself. With this function we have to think about some boundary conditions as well like if we pass empty list to this function, then the output is undefined because we cannot compute first of empty list.

(Refer Slide Time: 03:02)

last(l)



e.g. `last([20, 30, 40, 50, 10])` is 10

```
last(l) {  
    foreach x in l {  
        e = x  
    }  
    return(e)  
}
```


TUTORIAL ON PSEUDOCODE FOR LIST FUNCTIONS

4

Last is another function which returns the last element in the provided list. Here, foreach iterates over the elements in the list and will always update value of this variable E. Therefore when the

foreach block ends its execution variable `e` will hold the last value in the list. Similar to `first` function if we pass an empty list to the `last` function then the output is undefined.

(Refer Slide Time: 03:35)

 IIT Madras
ONLINE DEGREE

rest(l)

e.g. `rest([20, 30, 40, 50, 10])` is `[30, 40, 50, 10]`

```
rest(l) {  
    found = False  
    restList = []  
    foreach x in l {  
        if (found) {  
            restList = restList ++ [x]  
        }  
        else {  
            found = True  
        }  
    }  
    return(restList)  
}
```

rest(l) = l - first(l)

TUTORIAL ON PSEUDOCODE FOR LIST FUNCTIONS5

`Rest` is a special function which always returns a list. This list will contain all the elements from the original list except the first element. It can also be represented as `rest of l` equal to `l` minus `first of l`. In the pseudocode we will maintain an empty list called `rest list` which we will use to store output of this `rest` function. Also we will maintain a Boolean variable `found` and initialize it to `false`. This variable will be set to `true` only once when we access the first element in the list.

From there onwards we will copy all the elements from the original list to the `rest list`. If we pass an empty list to this function then the `foreach` block will not execute and the function will return an empty list. It will also return an empty list if we pass a list of length 1.

(Refer Slide Time: 04:45)

`init(l)`

e.g. `init([20, 30, 40, 50, 10])` is `[20, 30, 40, 50]`

```
init(l) {  
    found = False  
    initList = []  
    foreach x in l {  
        if (found) {  
            initList = initList ++ [prev]  
        }  
        else {  
            found = True  
        }  
        prev = x  
    }  
    return(initList)  
}
```

`init(l) = l - last(l)`

TUTORIAL ON PSEUDOCODE FOR LIST FUNCTIONS

6

Init function returns all the elements from the original list except the last element. Hence it can be represented as `init of l equal to l minus last of l`. The pseudocode is similar to `rest` function except the variable `prev` which is used to store the value of previous element while iterating over the list. No element will be added to the `init` list when variable `x` holds the first element of original list. If variable `x` has second element of the original list then we will be adding the first element in `init` list which is stored in variable `prev`.

The same process of adding an element will continue for all the elements of the original list except the last because we will add the second last element of original list in the `init` list when variable `x` has the last element of the original list. Similar like `rest` function, if we pass an empty list to this function then the `foreach` block will not execute and the function will return an empty list. It will also return an empty list if we pass a list of length 1.

(Refer Slide Time: 06:22)

`member(l, e)`



e.g. `member([20, 30, 40, 50], 30)` is True, `member([20, 30, 40, 50], 10)` is False

```
member(l, e) {  
    foreach x in l {  
        if (e == x) {  
            return(True)  
        }  
    }  
    return(False)  
}
```

TUTORIAL ON PSEUDOCODE FOR LIST FUNCTIONS

7

Member is another special function which checks for the membership of an element in a list. It accepts two parameters; a list and an element and returns Boolean value true or false based on whether the input is present in the list or not. If list `l` contains an element `e`, then the function will return true which will stop the iteration of the foreach block and member function will terminate.

If that return statement does not execute at all then it indicates that the list `l` does not contain the element `e`. In such case for each block will complete its execution without any output and then the default return statement will execute which will provide the output value false. Thank you for watching this tutorial, happy learning.