SEER AKADEMI
LINUX PROGRAMMING

AND SCRIPTING -
PERL 9

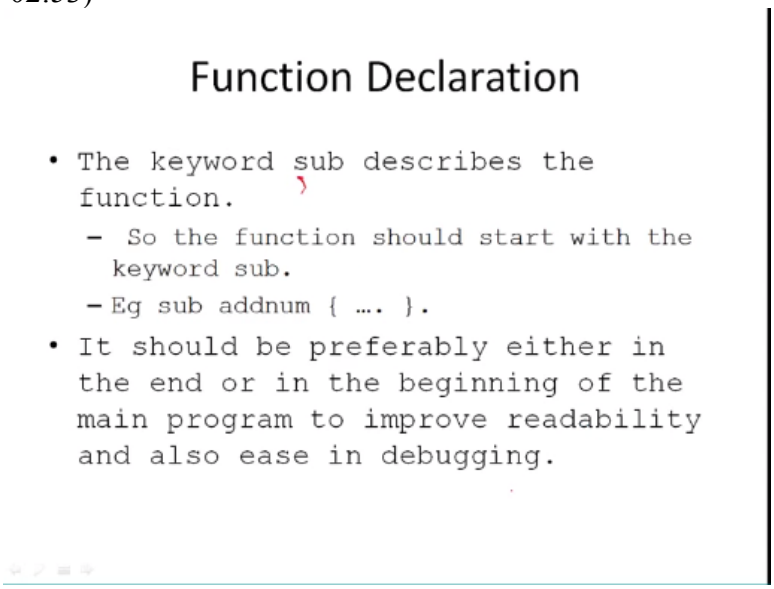
Oh other students again welcome to the class.
(Refer Slide Time: 00:11)

So this is about the Linux programming and scripting, today we will be talking more about the Perl ,today's class, in today's lecture, I am going to talk mostly about the things, that we have already talked about, I want to recap some of the concepts and also give you some more additional items with respect to Perl, as you know Perl is a very versatile language, used very widely across many applications, not just limited to the software side.
It is also used in the chip design, side of things lot of them using Perl as the main scripting language ,so last week ,we covered some of the things, about functions, I am going to go through it ,one more time just to import the concepts, we talked about the function declaration ,how to call a function ,passing the parameters, local variables and returning values ,so these are all the main things that we talked about.
I hope that you have already understood this just in case ,I am going to go through these things in more detail, so today we can think of today's session, is largely a recap session ,but there will be like few more items, that will be covered again those things like control structures, that we talked

about briefly ,when the address the data structures themselves ,we talked about in context of the data structure .

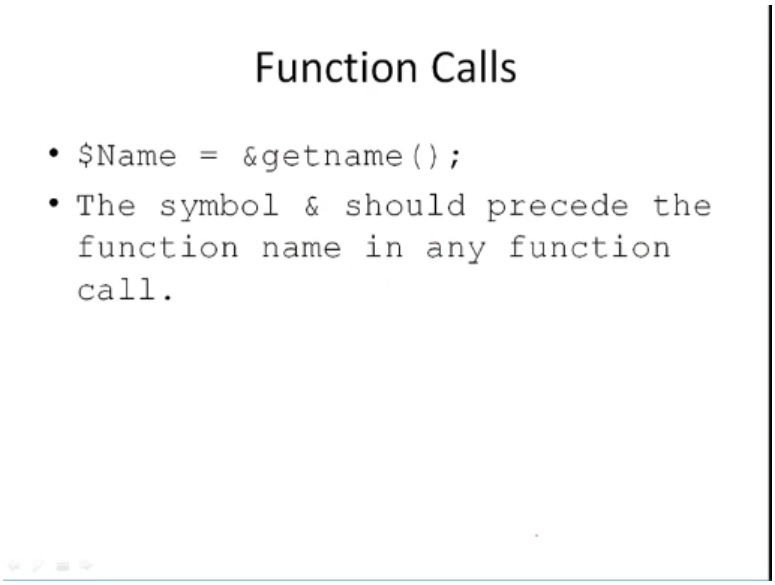
Here we will be more formalizing and talking about all the control structures work, what are the different control structures ,etc so again on the functions ,we talked about the function declaration ,follow in for function declaration ,we use the keyword stub to discuss the function , these parameters are required ,so the function should always start with the keyword sub ,it should preferably either in the end ,or in the beginning of the main program, this improves the readability.

(Refer Slide Time: 02:53)



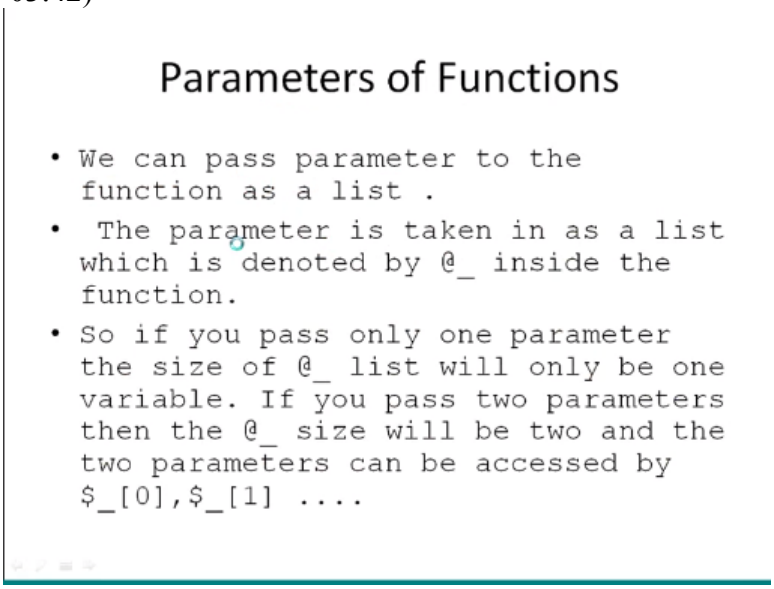
Perl does not have any such requirement, there the functions declarations should be, so you can you can use it, in the beginning of the program or at the end of the program, but use it in one place consistently, all the time, so that that improves readability.

(Refer Slide Time: 03:11)



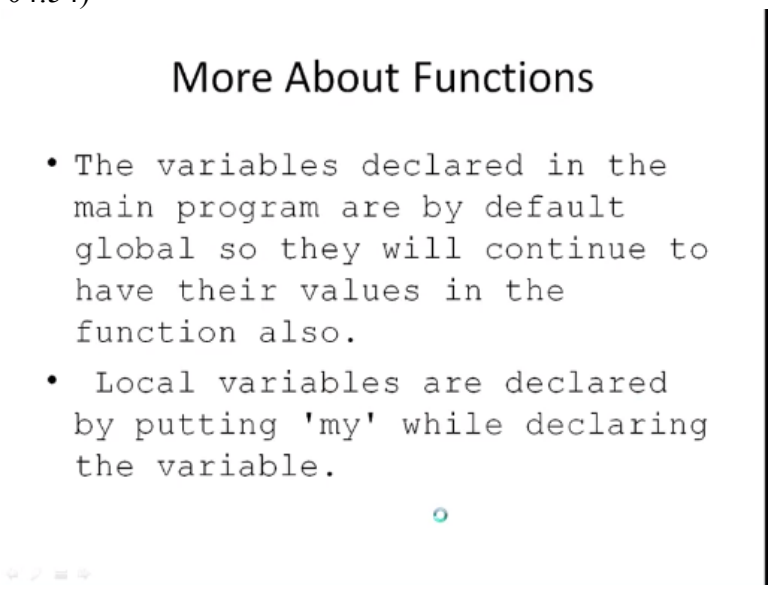
And then how do we use functions essentially, like that is function call ,so your get name is the function and then we provide this & and then once we specify that, becomes Perl understands that this is a function and it goes for the particular function with narration and then it looks for where the function is actually defined and then it runs the function, so the symbol & should always precede the function call in any function call.

(Refer Slide Time: 03:42)

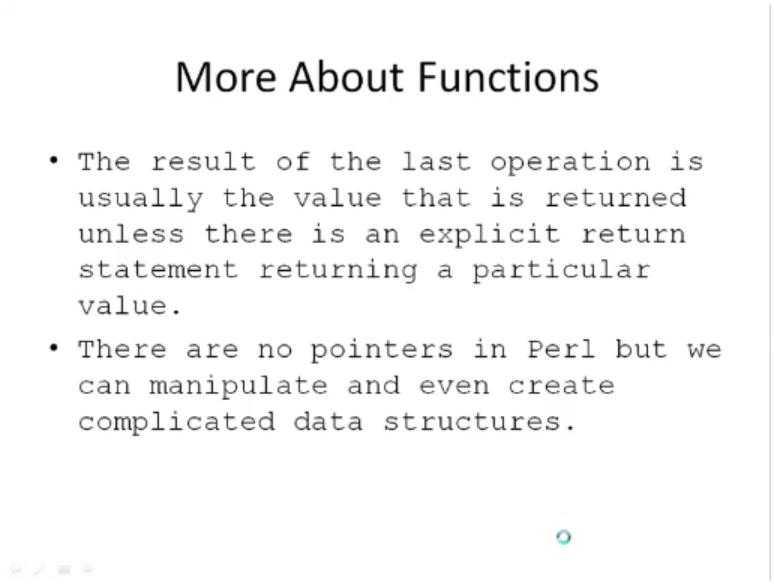


The parameter is essentially like I mean, always the parameters are passed as a list ,so even if you have array that you are passing , that is expanded and the internal list , expands parameter one of the thing about Perl is that, the Perl keeps this parameter list in this particular variable, which is an array variable , & or @_ and this essentially like I mean means that it is this list and then you can directly access, each of the parameter by accessing the elements of this list .

So if you pass only one parameter, the size of the this particular list is just one variable, but if you pass two parameters then these can be $_ [0], and $_ [1], and then if you have more number of parameters, this just goes on, in the array thing is a very nice feature.
(Refer Slide Time: 04:54)

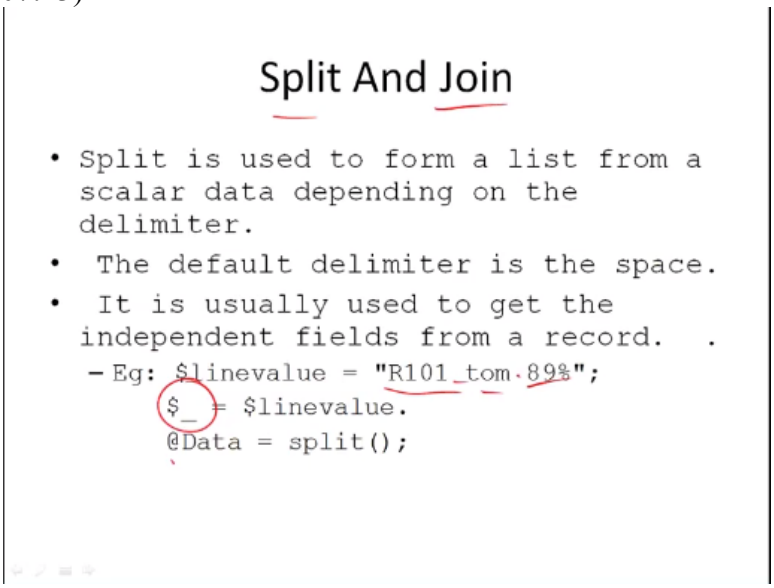


So that you always like I mean does not really constrain, you to have only like fewer number of variables, when performing call I mean performing a function part ,the variables declared in the main program are by default global variables, so they will continue to have their values in the function also, what this means is essentially the scope of the variable, is extended into, the extends into function ,okay so if there are any local variables , declared by using my in the beginning to declaring the variable .
So these my types of variables will have the context , specified in that level ,so if you are using my inside the functions and so that has a scope only, within that function does not go outside right off set of it .
(Refer Slide Time: 06:10)



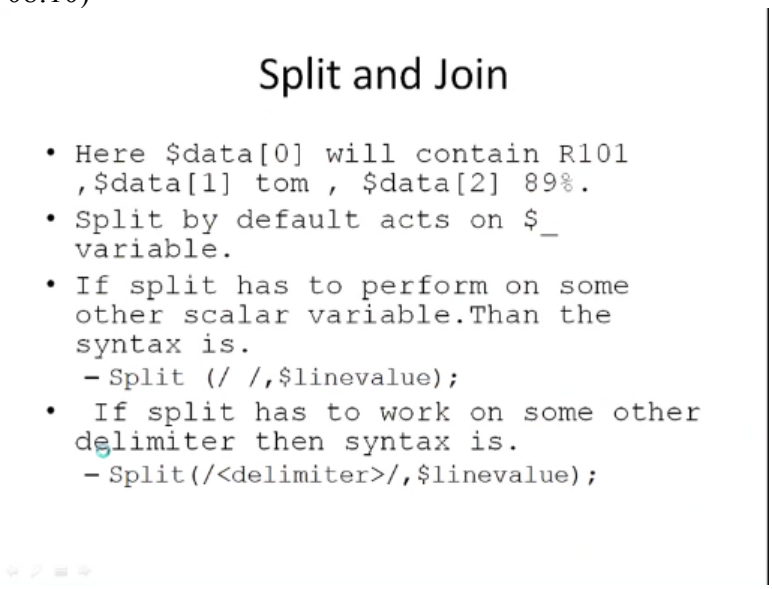
And we also saw the return values essentially ,becoming the result of the last operations with them, unless you have an explicit return statement ,in the function and in which case whatever the return value that specified inside the return statement ,that is returned ,one thing to notice there are no pointers in Perl, but we can manipulate and even create complicated data structures okay, so the other thing that we just touched upon was is used to form a list from a scalar data . Depending on the delimiter, so if so here one example is R 1 0 1 ,tom 89% and then we declare this as basically like an specific array .

(Refer Side Time: 07:13)

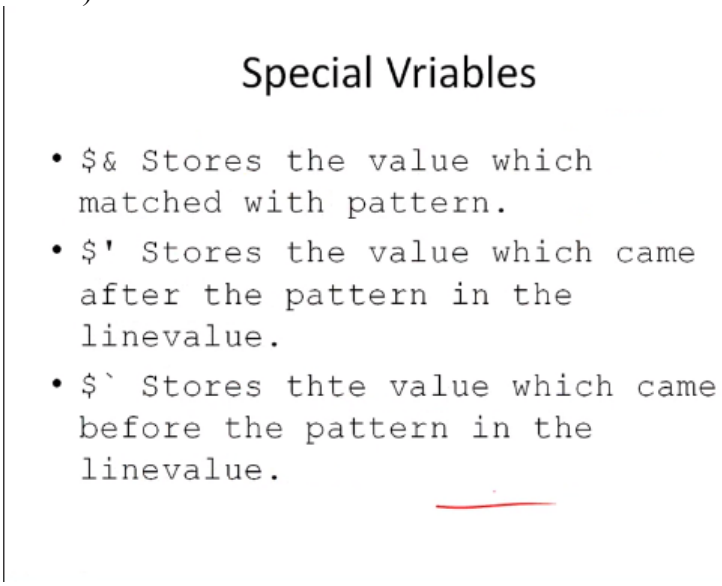


And then the split and then that means that basically, this split happens, the blank space here and it is on , this variable which is $_ so the resulting value is essentially an array like a data $ [0]= 1

0 1 $ data [1]=tom and then $ data [1] 89% ,so the default delimiter is the space and the default arrays is $.
(Refer Slide Time: 08:10)

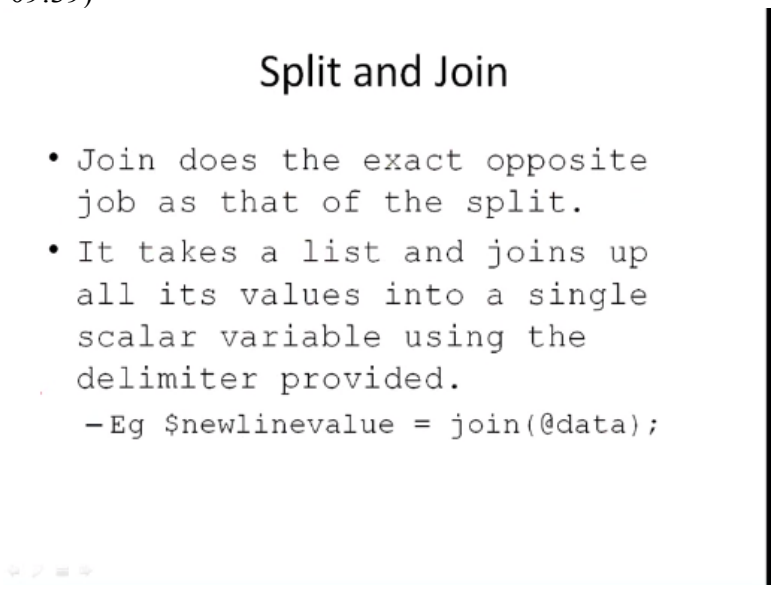


So the data actually should be $, so the data [0] will have 1 0 1 and then data [1] will have Tom and then data[2] we have 89% ,so as I mentioned the default is $_ variable, if you want to perform on another scalar variable then the syntax is split with the delimiter and then the actual behavior ,so here the delimiter is again still from one , but in general the syntax is the delimiter,$ line value , and then the variable itself ,there which one you have and skip .
(Refer Slide Time: 09:01)

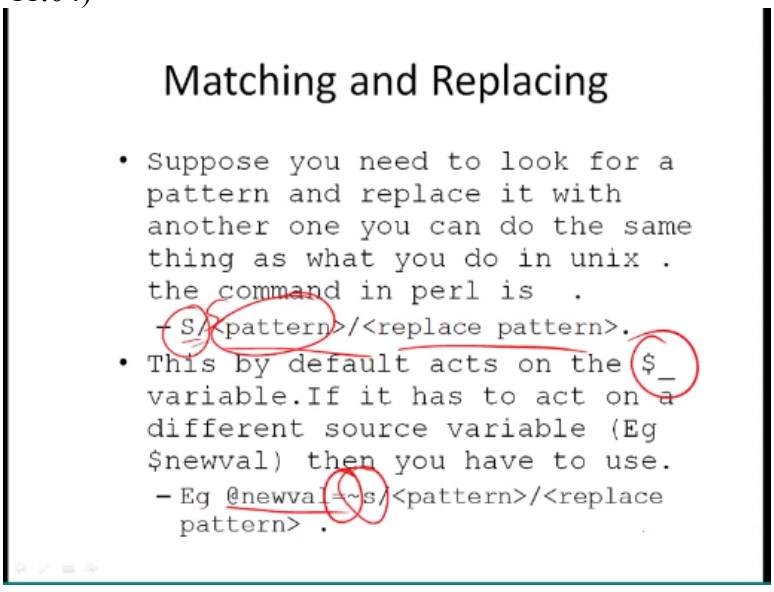


We also saw this in the last class ,if we have the script ,the $, & so the value of the matched pattern ,all this is it for the matching, for example ,and then this one the $ overtakes stores the

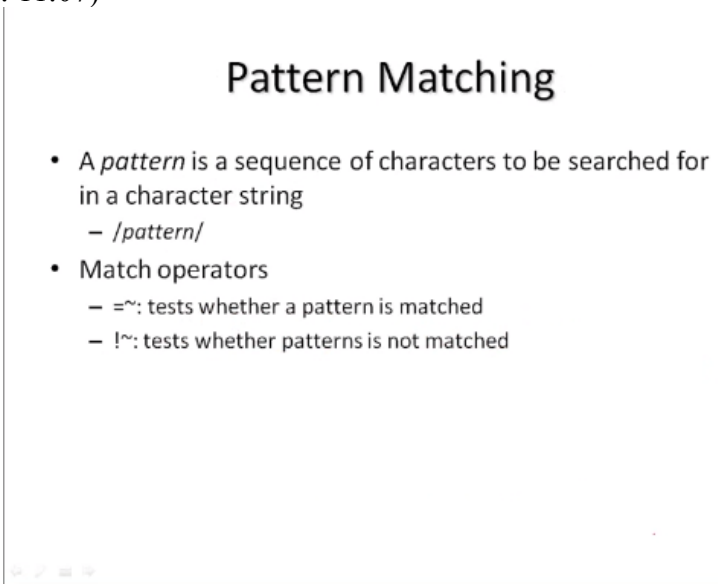
value that is after the pattern and then follow dollar back take stores the value that , before the pattern, so now let us look at the join, the join actually does exactly opposite job as split .
(Refer Slide Time: 09:39)



It takes a list and then joins up into all, its values into a single scalar variable and this also use delimiter provide you, now matching and replacing this is also something, that we saw in the last class or the last lecture, if you need to look for a pattern and replace it with another one, we can do it with this s command ,this is actually a lowercase s ,on this move down ,with the replacement pattern ,again by default this axis on the $_ variable you can act ,you can make it work with other variables .
And say like I mean okay in this new Val, this is what we want I want substitute this pattern with the replacement pattern.
(Refer Slide Time: 11:04)



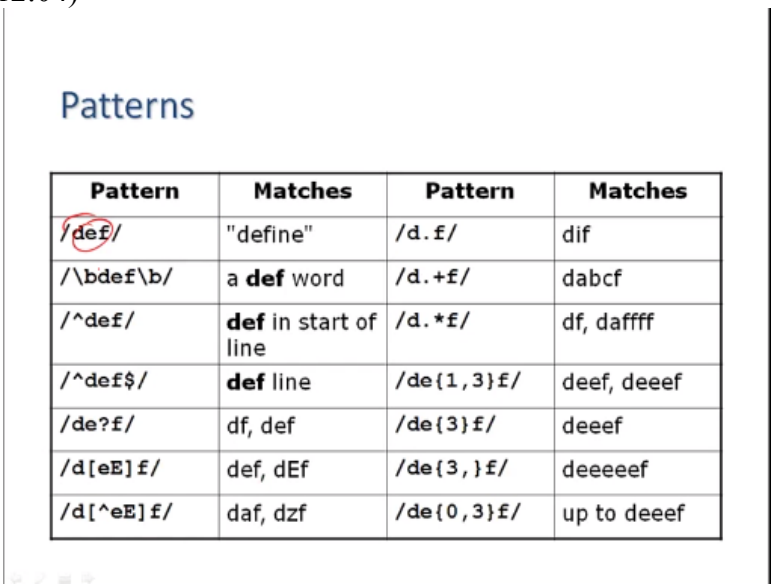
(Refer Slide Time: 11:07)



Now what is pattern and what is pattern matching this is something, that is crucial also like this is not, we did not talk about it in a big way, these lectures, the pattern is a sequence of characters that can be searched for in a character screen and usually the pattern is represented by the slash and , in between two slashes and then we know that = ~ :tests ,if the pattern is matched and there is also a way to do a test with the pattern is not matched and that is !~: ,so what are the patterns.

So here typical pattern and one pattern is for this the set of characters, it was within the slashes.

(Refer slide time: 12:04)



And this will match even defined, because we left part of the before ,so it little power match that ,if you specify two blank spaces, then it only matches the blank on either side of def ,here the character actually anchors the Def of the first three characters in the line, so here it matches this

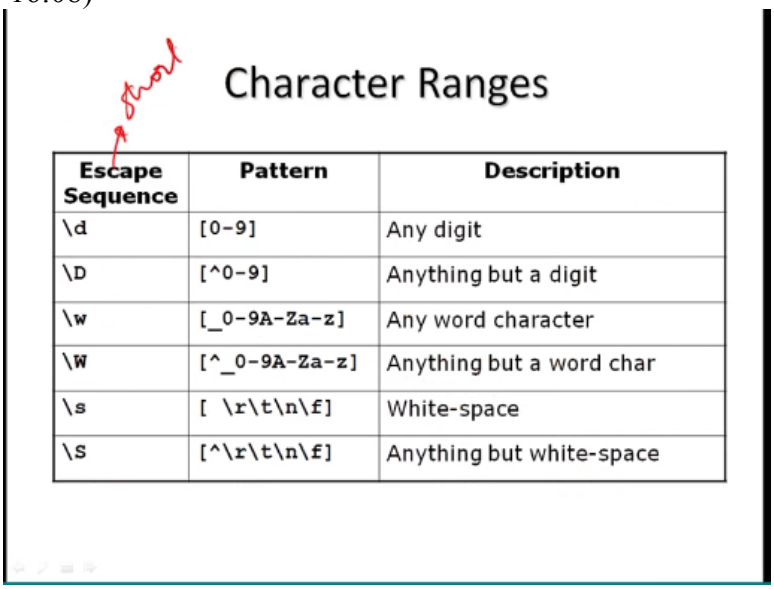
and then if you ,specify like I mean the $ denotes end of the line ,so if you are saying like character and Def and then dollar ,then it has only one word in the line, which is def .

So only the def will match and then the question mark sign is basically used for any in the middle , so here D question mark F matches def ligament ,so there is nothing here ,the is basically like it may or may not exist , now the d and then like in [e] or uppercase e this matches either def with the load is e or yet to see and then if you do a character , before the EE then if there is e ,then it won't match and anything other than E in Max .

For example a being everything, then if you put a dot and then that means that you can match ,any character in the middle, so it can be space ,non space in fact ,so all these things like that and then the + denote, basically it is 1 or more, so it is the a b c f will be matched and then if you put a * then that is zero or more ,so df will be matched and then the being match ,so all these things this is going to move, now this is an interesting one, it is a double braces on one and three and it is des ,what this means is the e can occur 1 to 3 times.

So essentially it can be you have def or the eee ,but in the double braces and then it will see ,if those many items are there and at 3 , the 2 empty spaces , 3 are involved, so it can match any number of e's, which is 3 variable and then 0 to 3 ,matches only up to def ,which is on 0 to 3 of E , so now let us look at the character ranges in themselves, as to how to specify and what do each of them denote , so in Perl we use like 1 2 3 4 5 6 different letters for denoting the pattern .So these are you can think of this escape sequence, as a short hand .

(Refer Slide Time: 16:08)



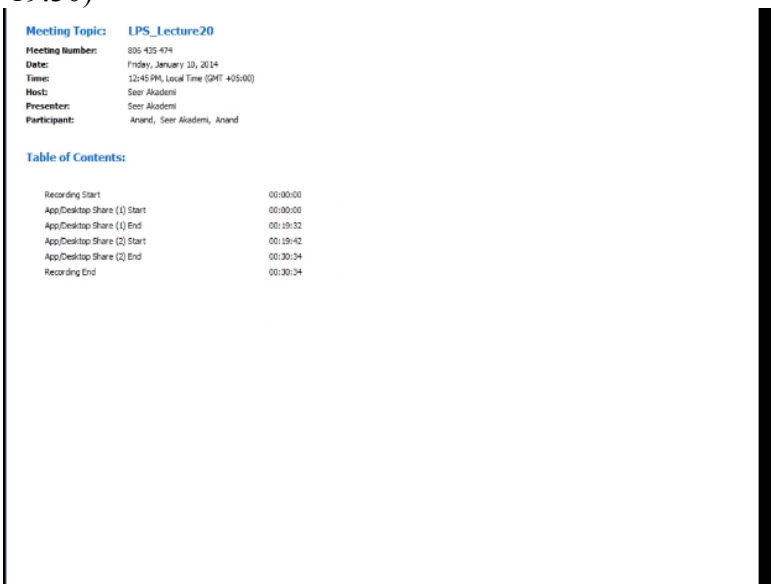
So these are you can think of this escape sequence, as a short hand notation ,so what I mean is the [0-9] means like any , if digit match anything , \ d , and then an uppercase D denotes

anything that is not digit ,which is same as putting the character in front of here , which is do not match , 0-9 but match anything other than that, now the W is essentially a word character which is an _ 0-9 ,uppercase A to Z and then for case a to z ,so all these are considered as a word characters.

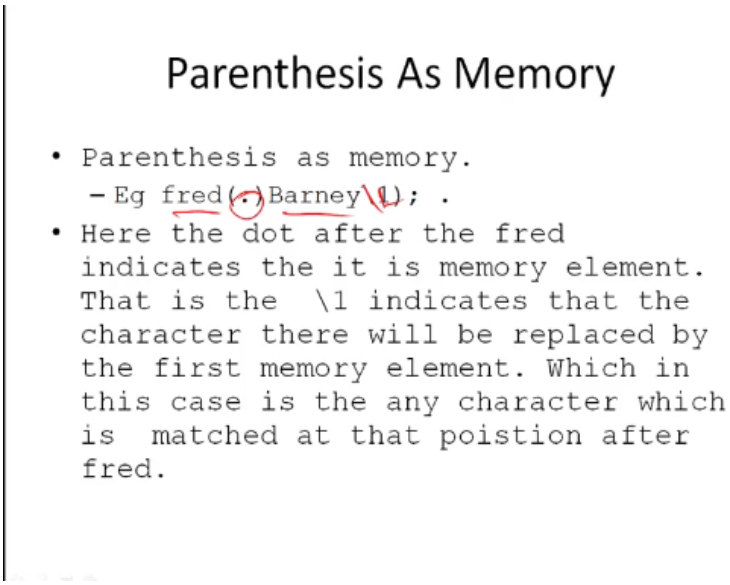
So if you specify \ w , that is the shorthand notation for this pattern ,so instead of that we can just use the \ w ,and then uppercase W is basically matches anything, but a word character, so that means that the same pattern ,with the character in front that denotes not of the pattern, \ f ,this is kind of counterintuitive here ,this is opposite or end of the file ,any non space characters unmatched using the \ s and then same thing like \ S .

We will match anything but and not once but a spaces , so that is why we have a character in front ,so these things have used you know sometimes, you can extract them out , so here there is a pattern it is Fred and Barney and then basically.

(Refer Slide Time: 19:30)



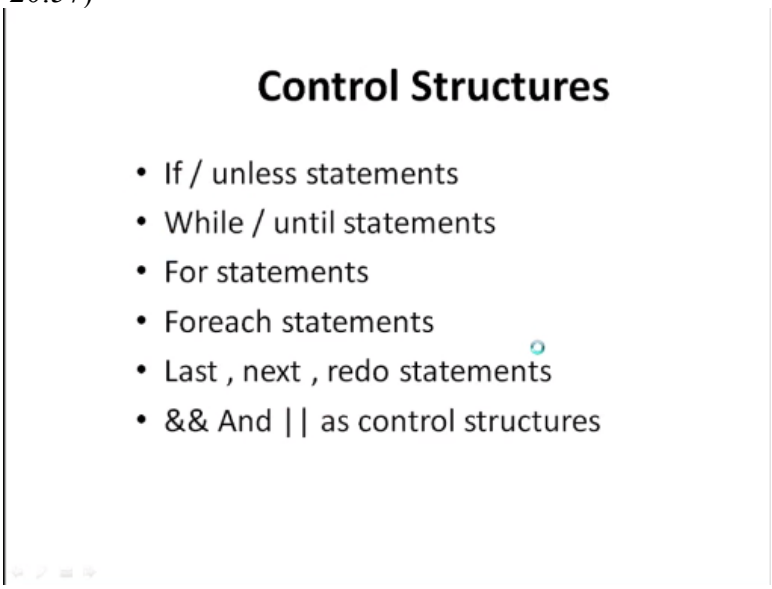
(Refer Slide Time: 19:48)



So as you saw actually , we can use the parentheses as the memory, so here when we match the fred, Barney and then basically you give a character in between and then when we say like this \ 1 ,the 1 represents the 1st parenthesis ,so that means that it is this one ,so any replacement that will happen ,will happen on this 1st character and then you can , put many other parentheses and then you can denote like which one that you want to replace this, with the character .

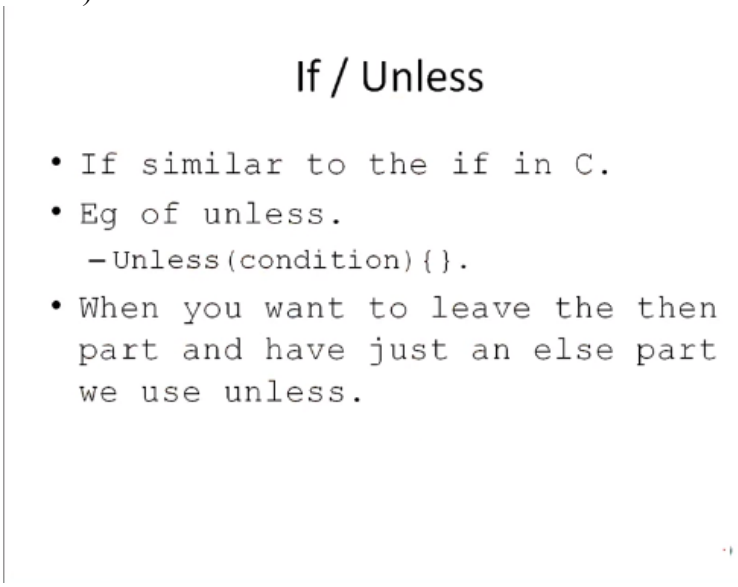
As I mentioned actually, this kind of using the parentheses as memory and we need to do the replacement, you can actually try it in vi, because it us a simple, similar syntax for doing the replacement.

(Refer Slide Time: 20:57)

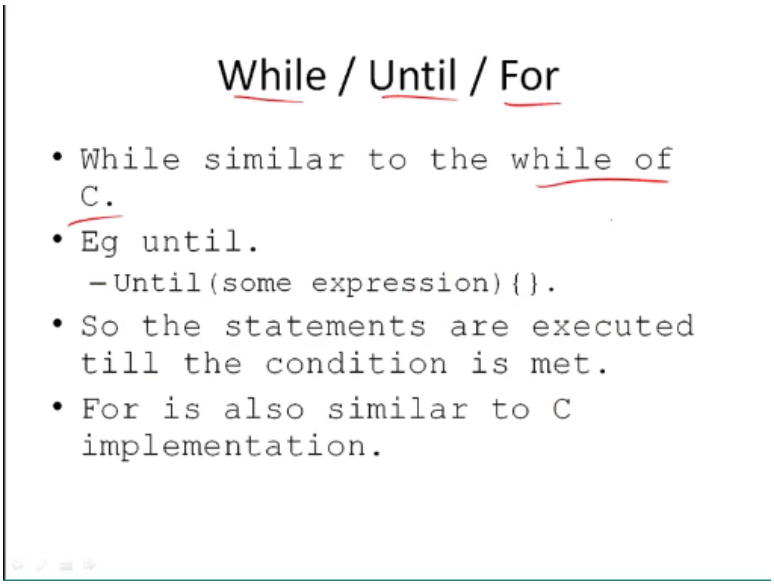


So let us talk about the control structures in Perl, we have the following control structures if ,unless statements, so if unless statements ,we will some examples ,how do we use these things

and then while ,until, the while is probably like I mean the most popular statements in Perl, then we also have for, these are formal for statements and then there are for each statements which can operate on , so we can do that .
And then we also have some of the control altering commands last, next and redo ,we will see how these things are used and then finally like I mean there is a logical operators actually become more like operators, but they can also be used as control structures ,and let us see how they are getting used.
(Refer Slide Time: 22:08)



So let us start with if ,the use of if is very similar ,to direct c , you already saw this ,with me in one of the earlier lectures ,the way we use if then followed the condition, then you open the {} and then write our program and then we close it and then we can say else if or else and then also put the condition , the unless is a command which also has very similar syntax as this and here the syntax is shown unless and then the condition and then the {}.
So when you want to leave the then part and then have just an else part ,we can just use unless, so this is basically, you have if and then else and we do not have anything here, then we want to use else and then with else of the same condition, basically instead of else , we use unless condition, okay so this is basically like for if and unless ,we will see a program with all these varies , constructs and then we will try to decipher what the program does.
And how do we document that , so and then the next one is while ,until and for the ,while is very similar to the while of c, usually like while is the one.
(Refer Slide Time: 24:04)



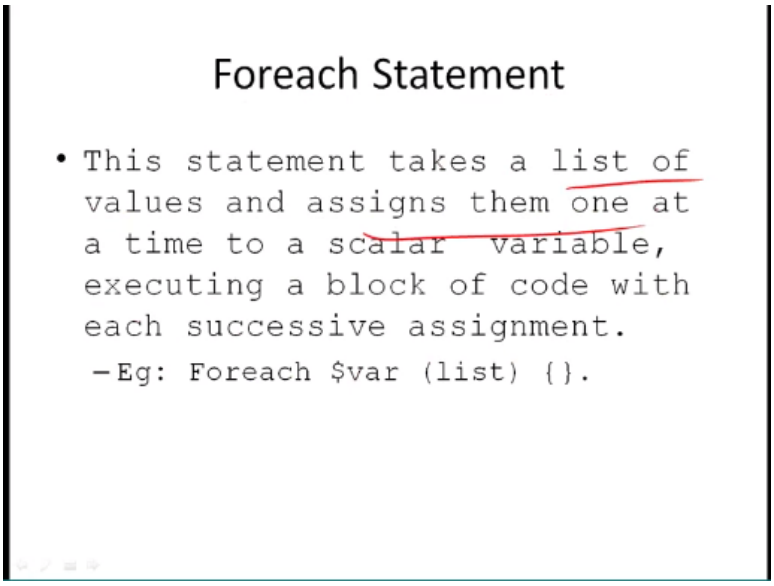
We use for loop ,so while some condition and then we use the {} to open closed ,for all the program elements here ,the while is kind of it is an infinite loop, as I mentioned this is essentially until the while, condition is satisfied, so for example if you are going to read a file , particular file ,we can always say while the file is still not empty, then we do some operations with the file line by line and then we do it and then the way to establish.

The while file is not empty is while as, there is the as long as, there are lines in the file ,so we just look for that condition basically like the $ line or the lines are still valid in the term , in the file , and then if we want to counter that basically and then that is we use until, so until essentially is in way , how we use and then followed by the condition and then the () and then the {}.

So here say like while condition and this is famous until not condition , so I think , so if and unless work the same way as while and until , so if is the equivalent to while, then unless is equivalent to until , so the statements are executed till the condition is met and then the for is also very similar to the C implementation, here for i =0 $ i and so this execute this loop executes 21 times or 20 times , so now the for each statement.

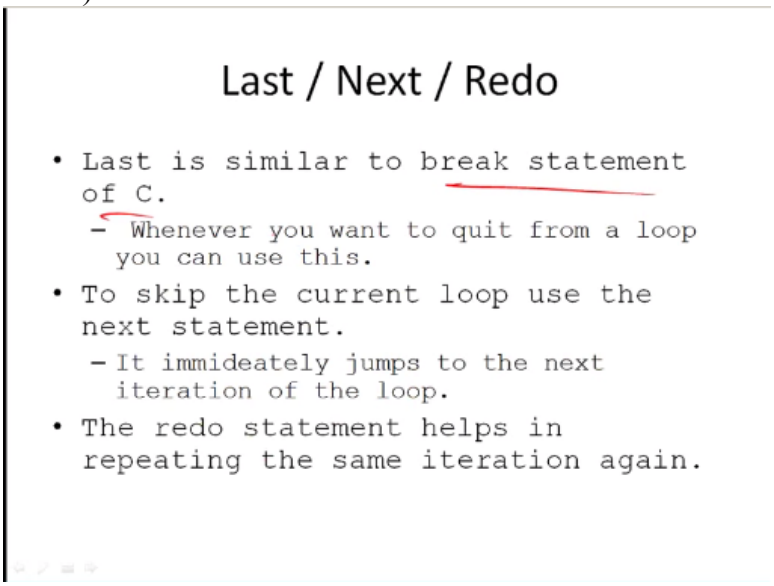
The four statement actually takes the list of values and assigns, them one at a time in a variable ,executing the block of the code with successive assignment , again if this is also a loop but here the argument is a list of values.

(Refer Slide Time: 27:48)



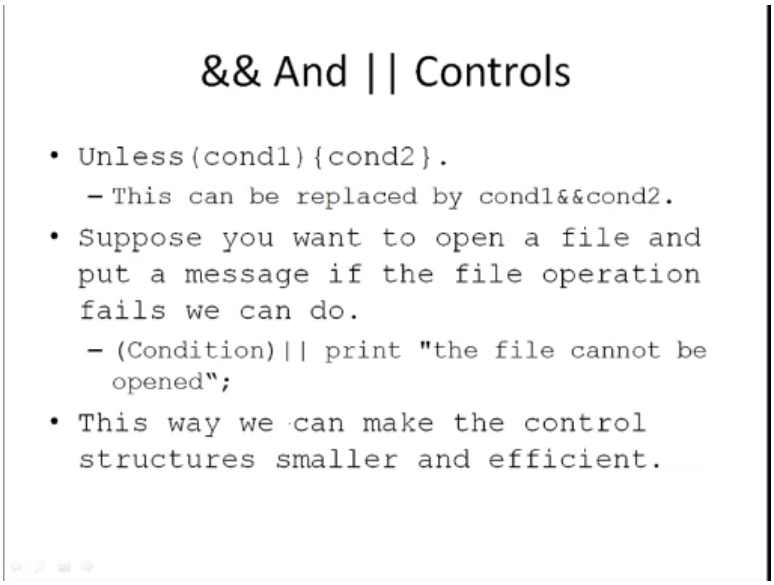
So we can say like for each $ var ,the list that means that the $ var ,will inherit list 1, list 2, list 3 etc and then we can perform operations on all the elements of that list , now let us look at the last is similar to the break statement of C ,so if you want to quit from the loop in the middle .

(Refer Slide Time: 28:30)



You will just use the last for and then if you want to skip ,the current loop and go to the next statement essentially ,we use in next statement and then this basically like I mean exist the current loop value, but still is in the loop and then boost next value and then keep going .

(Refer Slide Time: 29:10)



So we saw that the & has represented down the || and if you have a statement, unless condition 1 and then condition 2 you can just replace it ,by condition 1 and then condition 2 , because that is what the point and then if you, want to open a file and if you want to put a message that if the file operation is string , then we typically use the || operator to actually specify, that so we fail a condition and then || and then print the file ,cannot be opened , so when you use this kind of modification ,we make the control structure smaller and more efficient .