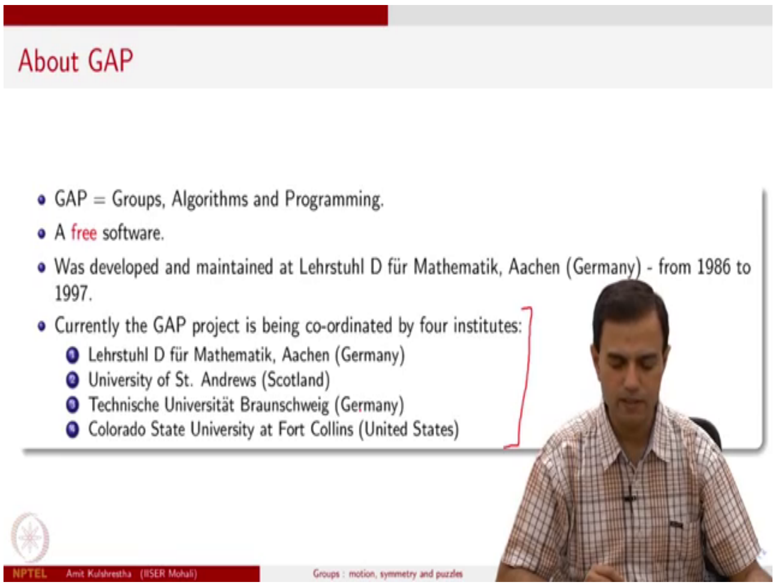
Groups: Motion, Symmetry & Puzzles

Prof. Amit Kulshrestha
Department of Mathematical Sciences
Indian Institute of Science Education and Research, Mohali

Symmetries and GAP exploration
Lecture – 11
Symmetries of plane and wallpapers

(Refer Slide Time: 00:14)



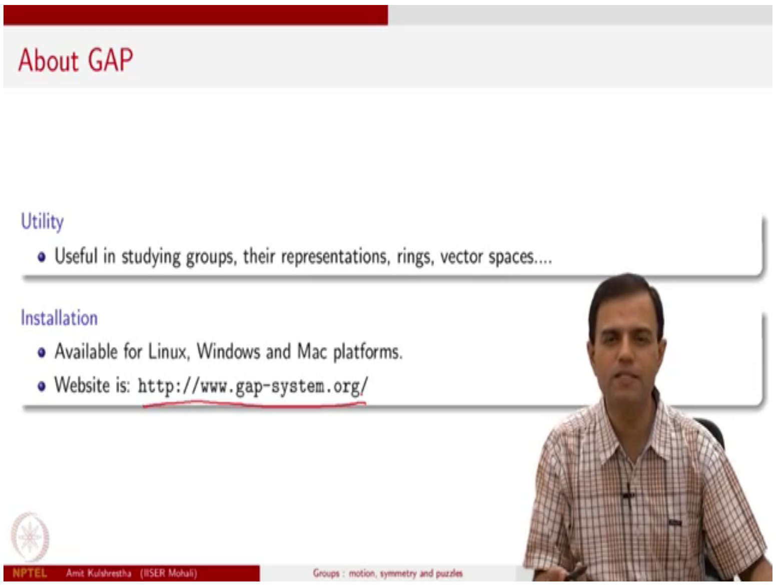
About GAP

- GAP = Groups, Algorithms and Programming.
- A free software.
- Was developed and maintained at Lehrstuhl D für Mathematik, Aachen (Germany) - from 1986 to 1997.
- Currently the GAP project is being co-ordinated by four institutes:
  1. Lehrstuhl D für Mathematik, Aachen (Germany)
  2. University of St. Andrews (Scotland)
  3. Technische Universität Braunschweig (Germany)
  4. Colorado State University at Fort Collins (United States)

NPTEL Amit Kulshrestha (IISER Mohali) Groups : motion, symmetry and puzzles

So, last time I had mentioned that we are going to learn how to solve Rubik's cube using gap. So, GAP stands for groups algorithms and programming it is a software and it is a free software. You can use it in various operating systems, windows, Linux, Mac and this software was maintained earlier at Aachen Germany, Aachen and currently there are 4 places all across the wide where the software is the this project is a being coordinated through. So, Aachen Braunschweig in Germany, Saint Andrews in Scotland and Fort Collins in United States, the 4 places are maintaining this software.

(Refer Slide Time: 01:14)


About GAP

Utility
- Useful in studying groups, their representations, rings, vector spaces....

Installation
- Available for Linux, Windows and Mac platforms.
- Website is: http://www.gap-system.org/

NPTEL Amit Kulshrestha (IISER Mohali)
Groups : motion, symmetry and puzzles

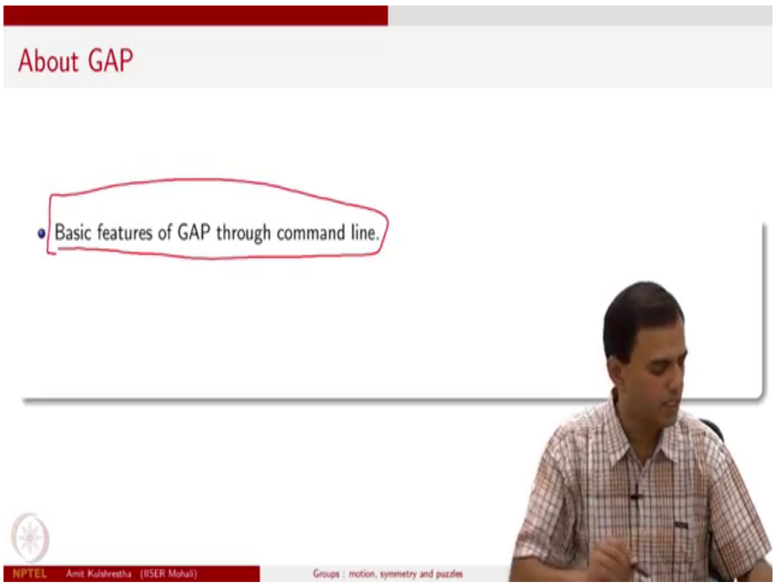
So, what is the specialty of GAP? Specialty of GAP is there it is, so it is a specialty of GAP specialty of GAP is that it is useful in studying groups. There is so many programming languages the traditional ones usual ones they have their own data structures they have their own structures and not many of them do support groups. So, here you can actually define groups you can play with groups you can define maps between groups and two calculations which otherwise they would be doing over a piece of paper. So, it is useful in studying groups.

Representations, I have not mention what representations are. So, far probably in coming lectures bit of representations what we are going to see, but as a rough way of understanding representations will be right the same group in (Refer Time: 02:16) itself, in various forms in various examples in various different real life situations. So, Rubik's cube as I said it is its representing something, its representing it is a tangible representation in sense something that you can touch it is a representation of its a group Rubik's group and you are going to talk about to Rubik's group as well.

But, in general representations concerned expressing elements of groups as if they are doing something, they could be mattresses, they could be permutations or maybe some other maps that could be relevant in certain context. So, they are going to learn about representations. Rings, probably some of you know, there are objects with two operations and there is a compatibility between those two operations practice this is most of you would know. So, all these algebraic structures can be quite nicely understood via the software called GAP. And as I said if your machine is Linux very good otherwise on windows and MAC platforms as well we will install the software.

And the website of GAP, GAP dash systems at all its a free software and it is a free in various senses, you can freely distribute it, you can make changes, you can add, you can contribute to it, you can make certain packages and donate it to the GAP systems and its all free of cost.

(Refer Slide Time: 03:58)



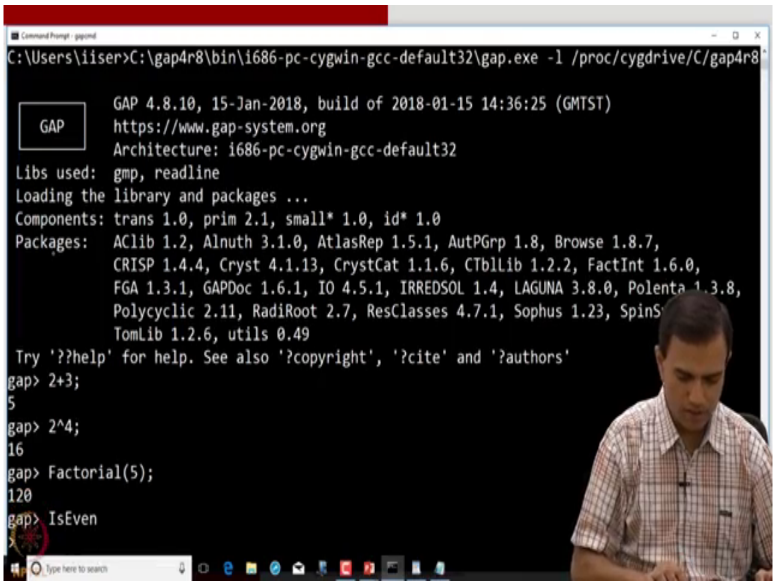
About GAP

- Basic features of GAP through command line.

NPTEL Amit Kulshrestha (IISER Mohali) Groups : motion, symmetry and puzzles

So, let us see what GAP can do for us. So, the command line through which GAP runs quite nicely.

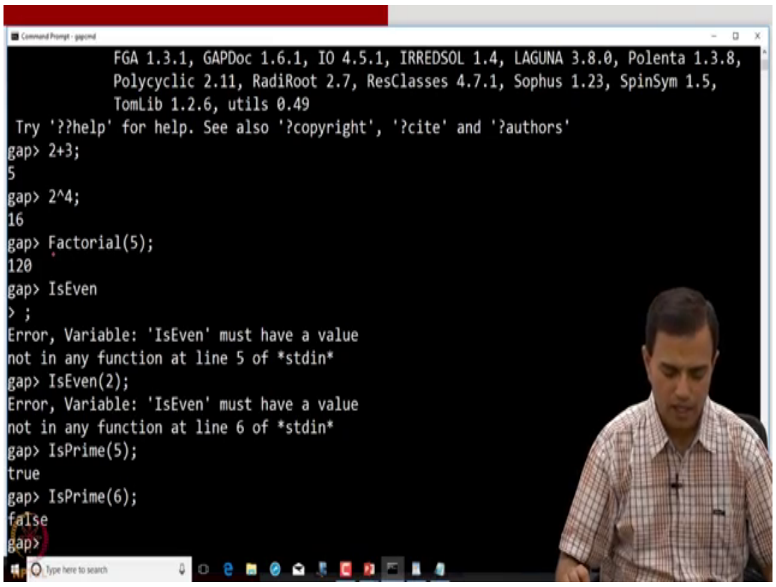
(Refer Slide Time: 04:09)



```
C:\Users\iiser>C:\gap4r8\bin\i686-pc-cygwin-gcc-default32\gap.exe -l /proc/cygdrive/C/gap4r8

 ┌───────┐   GAP 4.8.10, 15-Jan-2018, build of 2018-01-15 14:36:25 (GMTST)
 │  GAP  │   https://www.gap-system.org
 └───────┘   Architecture: i686-pc-cygwin-gcc-default32
 Libs used:  gmp, readline
 Loading the library and packages ...
 Components: trans 1.0, prim 2.1, small* 1.0, id* 1.0
 Packages:   AClib 1.2, Alnuth 3.1.0, AtlasRep 1.5.1, AutPGrp 1.8, Browse 1.8.7,
             CRISP 1.4.4, Cryst 4.1.13, CrystCat 1.1.6, CTblLib 1.2.2, FactInt 1.6.0,
             FGA 1.3.1, GAPDoc 1.6.1, IO 4.5.1, IRREDSOL 1.4, LAGUNA 3.8.0, Polenta 1.3.8,
             Polycyclic 2.11, RadiRoot 2.7, ResClasses 4.7.1, Sophus 1.23, SpinS
             TomLib 1.2.6, utils 0.49
 Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap> 2+3;
5
gap> 2^4;
16
gap> Factorial(5);
120
gap> IsEven
```

So, here is GAP, this is how the GAP screen looks like. And you can do various calculations on this. Many calculations are as usually networking were doing in other programming languages. For example, as basic as this I am just saying 2 plus 3 2 plus 3 right, we will tell you the answer. And there are other commands as well for example, let us see 2 to the power 4 it will tell you the answer quite easy factorial say 5 and this is something after every command you have to give the semicolon factorial 5. Oh yes, it does recognize factor in 5 and then there are many in built commands.

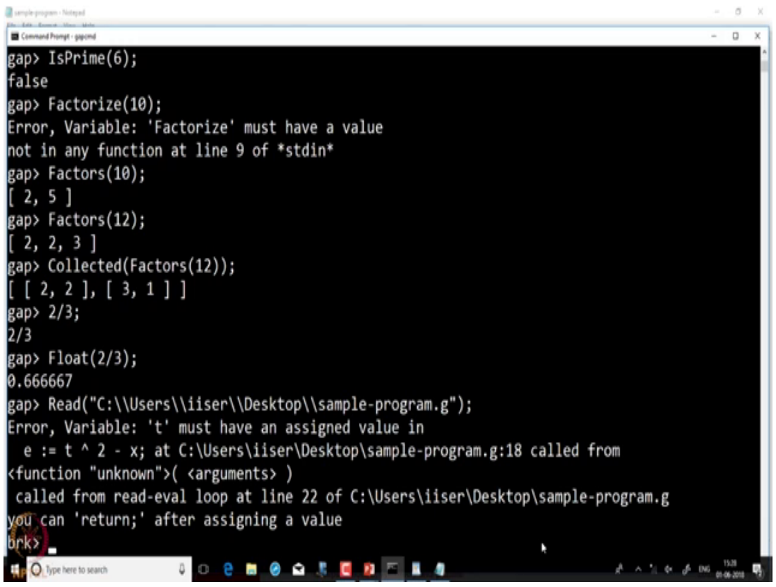
(Refer Slide Time: 05:17)



```
             FGA 1.3.1, GAPDoc 1.6.1, IO 4.5.1, IRREDSOL 1.4, LAGUNA 3.8.0, Polenta 1.3.8,
             Polycyclic 2.11, RadiRoot 2.7, ResClasses 4.7.1, Sophus 1.23, SpinSym 1.5,
             TomLib 1.2.6, utils 0.49
 Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap> 2+3;
5
gap> 2^4;
16
gap> Factorial(5);
120
gap> IsEven
> ;
Error, Variable: 'IsEven' must have a value
not in any function at line 5 of *stdin*
gap> IsEven(2);
Error, Variable: 'IsEven' must have a value
not in any function at line 6 of *stdin*
gap> IsPrime(5);
true
gap> IsPrime(6);
false
gap>
```

Let us see is even sorry is even 2 there is no such command. Let us see is prime is that a command is prime 5, oh yeah there is a command is prime is prime 6 no there is no such command. So, the usual things it does all the usual calculations it does. Let us see if it can factorize something.

(Refer Slide Time: 05:46)



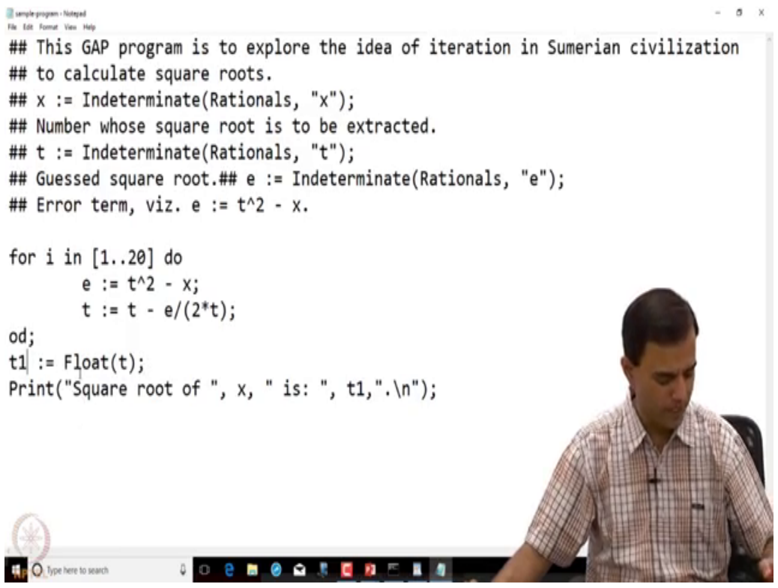
```
gap> IsPrime(6);
false
gap> Factorize(10);
Error, Variable: 'Factorize' must have a value
not in any function at line 9 of *stdin*
gap> Factors(10);
[ 2, 5 ]
gap> Factors(12);
[ 2, 2, 3 ]
gap> Collected(Factors(12));
[ [ 2, 2 ], [ 3, 1 ] ]
gap> 2/3;
2/3
gap> Float(2/3);
0.666667
gap> Read("C:\\Users\\iiser\\Desktop\\sample-program.g");
Error, Variable: 't' must have an assigned value in
  e := t ^ 2 - x; at C:\Users\iiser\Desktop\sample-program.g:18 called from
<function "unknown">( <arguments> )
 called from read-eval loop at line 22 of C:\Users\iiser\Desktop\sample-program.g
you can 'return;' after assigning a value
brk>
```

Factorize all the also the command names are quite usually easy to remember factorize say 10, it does not maybe factors 10 says 2 and 5 factors 12, 2 to 3. So, it does it and then this is nice command for collected, so collected factors 12. So, during converted into bunches. So, there are 2 to the power 2 times 3. So, 2 to the power 2 times 3.

So, first is the integer and 2 is the; is exponent. So, integer 2 to the power exponent times another integer to the power exponent 2 to the power 2 times 3 to the power 1. So, like that you can do all the things which usually you could do with the various programming languages. You can run loops also with this. So, let us let us see few more commands, let us try to divide 2 divided by 3 it simply says 2 by 3.

So, it does not try to approximate things, it does not try to put things in decimal fraction because sometimes when computer put certain expression in the decimal form there is an error involved. So, GAP tries to be as accurate as possible. Let us see what kind of programs it can run. So, first we are going to see programs which are not quite group theory programs.

(Refer Slide Time: 07:33)



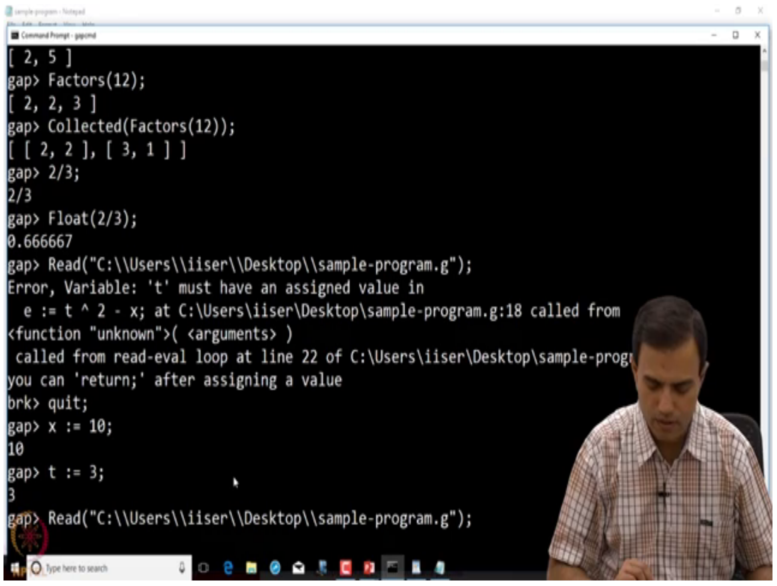
```
## This GAP program is to explore the idea of iteration in Sumerian civilization
## to calculate square roots.
## x := Indeterminate(Rationals, "x");
## Number whose square root is to be extracted.
## t := Indeterminate(Rationals, "t");
## Guessed square root.## e := Indeterminate(Rationals, "e");
## Error term, viz. e := t^2 - x.

for i in [1..20] do
       e := t^2 - x;
       t := t - e/(2*t);
od;
t1 := Float(t);
Print("Square root of ", x, " is: ", t1,".\n");
```

The sample program what does it do? It just calculates this program runs the square root of say this case square root of 2 or square square root of any given number x. The algorithm which is there is a quite classic this is the algorithm which was there in Sumerian civilization much (Refer Time: 08:03) much before now more than 2-3000 years.

So, what it does is it urns a loop, so for certain loops supposed to certain things those who are familiar with programming would understand what is happening this is usually, this is usually in any programming language. So, you start with a guessed square root. So, x is what for which you are looking for square root t is something this guessed square root some guess initial value and e is the error and in each term you improve your guess by this. Now, the point is why should it work and that is genius of Sumerian people which is not very difficult for you to work out that why this way of iterating this way of assigning new value to your guessed value of the square root is writing to do. So, and then just write it in the decimal form float t.

So, what does flow do? Let us go back there and see, float 2 by 3 let us see what it does. So, it does, but it has a it has a output you are given input in the decimal format. And it is a sample program. So, let us see what is the output of this and I am iterating it 20 times. So, I am just giving the path of the sample program. So, this is a way you write your program and the file that I had shown you was here sample program dot g, and this is the path of the file and now I am going to execute it. So, one way to execute is to read it like this read within codes you put the file, file name, the whole path and see what it does. So, I am going to enter there is an error, what should have been the error. Let us go back to the program and see x is something, what is x? It does not know. What is t? It does not know. So, we have to ask whose square root we want and then what is our guessed value of the square root. So, let us go back and correct that error.

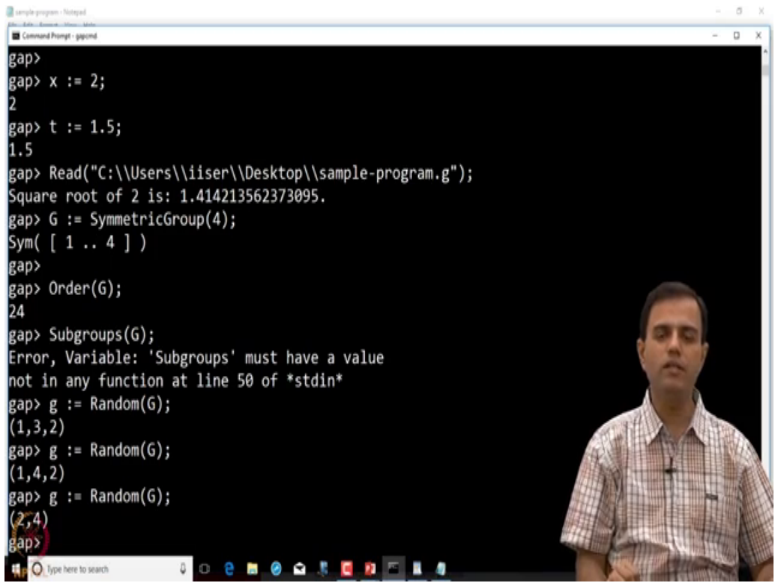
(Refer Slide Time: 10:45)



```
[ 2, 5 ]
gap> Factors(12);
[ 2, 2, 3 ]
gap> Collected(Factors(12));
[ [ 2, 2 ], [ 3, 1 ] ]
gap> 2/3;
2/3
gap> Float(2/3);
0.666667
gap> Read("C:\\Users\\iiser\\Desktop\\sample-program.g");
Error, Variable: 't' must have an assigned value in
  e := t ^ 2 - x; at C:\Users\iiser\Desktop\sample-program.g:18 called from
<function "unknown">( <arguments> )
 called from read-eval loop at line 22 of C:\Users\iiser\Desktop\sample-prog
you can 'return;' after assigning a value
brk> quit;
gap> x := 10;
10
gap> t := 3;
3
gap> Read("C:\\Users\\iiser\\Desktop\\sample-program.g");
```

So, now it is in some break loop, I will come out of it by writing quit, I do that and now let me define x. So, this is a way to define x colon is equal to say I want this square root of 10 and let me say the guessed value t is that me say 3, and then I run the program. Then I just call this program using read command and see what is happening. So, it has gone into a loop this time.

(Refer Slide Time: 11:21)



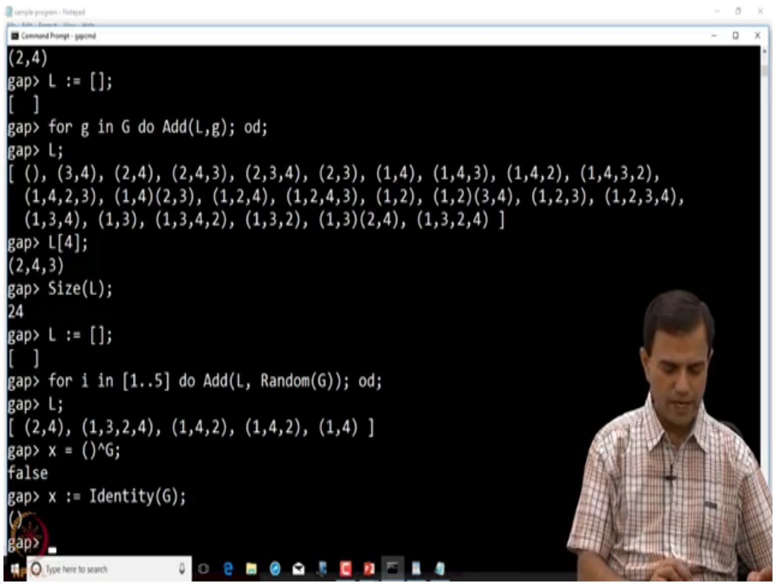
```
gap>
gap> x := 2;
2
gap> t := 1.5;
1.5
gap> Read("C:\\Users\\iiser\\Desktop\\sample-program.g");
Square root of 2 is: 1.414213562373095.
gap> G := SymmetricGroup(4);
Sym( [ 1 .. 4 ] )
gap>
gap> Order(G);
24
gap> Subgroups(G);
Error, Variable: 'Subgroups' must have a value
not in any function at line 50 of *stdin*
gap> g := Random(G);
(1,3,2)
gap> g := Random(G);
(1,4,2)
gap> g := Random(G);
(2,4)
gap>
```

So, here I am giving some value of x say x is a 2 and t say just some guessed value of the square root say 1.5 and I am calling the file program file like this and see what happens. So, it actually guessed the square root after 20 iterations 1.4142 or something and these are the program. So, all the usual things that happen in the programming language there, they are looping it there and let us see what is there for groups. So, how can we make GAP understand, GAP remember how it should store groups.

So, here is nice way I am defining groups. So, G is a say there are in built groups which it knows. So, G is symmetry group say 4. So, this is going to store in a variable called G switched group on 4 later that is S 4, which has 24, just 24 ready mix. So, it has stored something and then all the interesting things. So, order of G how many elements are there in G it will tell you how many elements are there, the 24 elements. And what other things can you do let us see sub groups of G there is no such function, I can pick a random element from this random G.

So, it will pick some random element from this. So, it has just picked some random element 1 3 2, once more I do some other element, once more I do some other elements. Interesting thing is that you can loop over the group. So, how does it do?

(Refer Slide Time: 13:40)



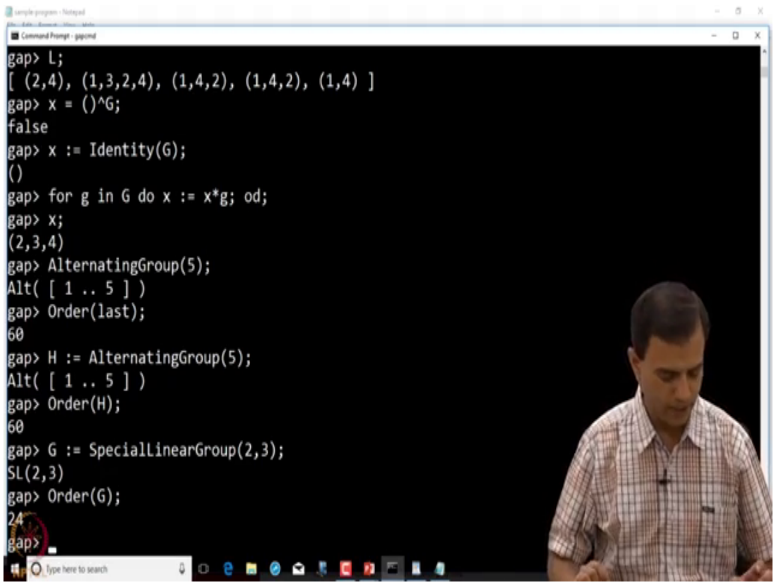
```
(2,4)
gap> L := [];
[  ]
gap> for g in G do Add(L,g); od;
gap> L;
[ (), (3,4), (2,4), (2,4,3), (2,3,4), (2,3), (1,4), (1,4,3), (1,4,2), (1,4,3,2),
  (1,4,2,3), (1,4)(2,3), (1,2,4), (1,2,4,3), (1,2), (1,2)(3,4), (1,2,3), (1,2,3,4),
  (1,3,4), (1,3), (1,3,4,2), (1,3,2), (1,3)(2,4), (1,3,2,4) ]
gap> L[4];
(2,4,3)
gap> Size(L);
24
gap> L := [];
[  ]
gap> for i in [1..5] do Add(L, Random(G)); od;
gap> L;
[ (2,4), (1,3,2,4), (1,4,2), (1,4,2), (1,4) ]
gap> x = ()^G;
false
gap> x := Identity(G);
()
gap> 
```

So, for g in G do now let us ask it to do something. So, well first let me make a list. So, n is empty list and now I can add in this for g in G do add to this list L, the element g and then I just close the loop. So, do is do and od is just closing the (Refer Time: 14:29).

So, it has put in this list called L all the elements of g is not it, it is a quite intuitive command and then I see what is L. So, it lists everything. And I want to see the 5th or 4th element of in the in the list it is going to 2 4 3, so 1 2 3 4 that is how it is showing. What is the size of L? Of course, 24. So, you can store elements in this or if you want say 5 random elements from this group, you can store you make a list I am just emptying the list is empty list and then I can do for g in G or rather for I in 0, 4 or say 1 5 do. So, it is making a list. So, so from the list, what are there in the 1 2 3 4 5th place it is going to store it.

So, do add to the list L some random element say random G, od and then we can see these are some 5 random elements from this. So, it can do all these kinds of manipulations. And one can multiply all the elements of the group for. So, I will start with say x is equal to this. So, this is notation for all this some problem this is a (Refer Time: 16:32) would be some something will I get it will I get it later So, I just want the product of all the elements. So, what do I do? I first store the identity element. So, let me call it x is equal to identity of g let us identity element of g.

(Refer Slide Time: 17:06)



```
gap> L;
[ (2,4), (1,3,2,4), (1,4,2), (1,4,2), (1,4) ]
gap> x = ()^G;
false
gap> x := Identity(G);
()
gap> for g in G do x := x*g; od;
gap> x;
(2,3,4)
gap> AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> Order(last);
60
gap> H := AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> Order(H);
60
gap> G := SpecialLinearGroup(2,3);
SL(2,3)
gap> Order(G);
24
gap>
```

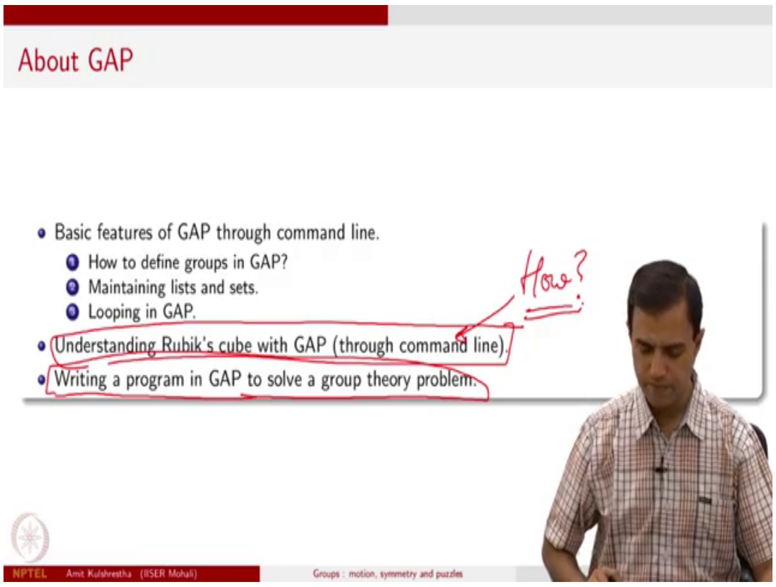
And I do for g in G do x is equal to x into g. So, it is multiplying all the elements of the group. So, started identity and in identity you keep adding all the elements do this and then I close it. So, the variable x in the end is going to store the product. So, in this case the product is 2 3 4, ok. So, just some fun things these are, there are some inbuilt examples of groups which are stored in this apart from alternating apart from symmetric groups you have what are called alternating groups, alternating groups a or alternating group on 5 letters and of course, you can check the order of this order auto meaning number of elements in this.

So, order in this order, so last. So, this command is useful whatever is the last output does it like this, or I could have just done it like H is alternating group on 5 letters and then order of H is this. So, you can play with the groups there are more inbuilt groups for example, yes I can matrix groups, G is a special linear group of 2 by 2 matrices with say special linear group, 2 by 2 size and over field of 3 elements. So, it stores it for you, what does order of the group it will output it. So, all these interesting things it does.

And now we are going to understand how we can use it to solve Rubik's cube. So, first of all how to tell this software this GAP, what is Rubik's group because there are so many moves right, there are so many moves here and everything is essentially permutation right. So, that is where we are going to use the idea of generators and relations we will not care much, but at least generators. So, you see when I am having

Rubik's cube here its being generated by these kind of moves, moves in the left, moves my top, moves in the in a right, bottom, front and back. So, there are rotations by 90 degree right, all these are rotations by 90 degree, I am rotating only one layer at a time top bottom or front like that. So, let us see let us go back and see how Rubik's cube be stored in this. So, these are some basic features.

(Refer Slide Time: 20:39)



About GAP

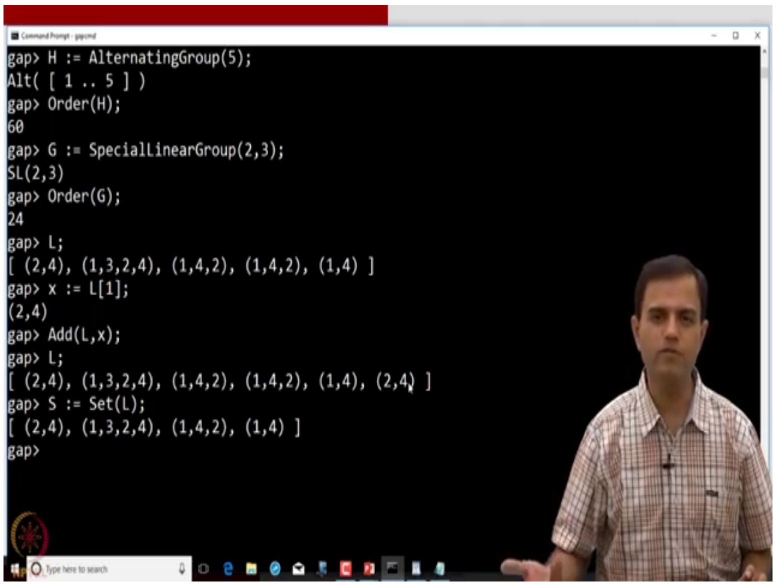
- Basic features of GAP through command line.
  1. How to define groups in GAP?
  2. Maintaining lists and sets.
  3. Looping in GAP.
- Understanding Rubik's cube with GAP (through command line).
- Writing a program in GAP to solve a group theory problem.

How?

NPTEL Amit Kulshrestha (IISER Mohali) Groups : motion, symmetry and puzzles

We have seen how to maintain lists we have seen that the GAP can do looping and we can define certain groups. As you read the GAP manual you will realize then there are other base there are so many other groups also and same groups also you can store in the group in the in the GAP in different ways sometimes in terms of generators and radiations as well. So, maintaining lists I have told sets just quickly. I will say take a list L.

(Refer Slide Time: 21:14)



```
gap> H := AlternatingGroup(5);
Alt( [ 1 .. 5 ] )
gap> Order(H);
60
gap> G := SpecialLinearGroup(2,3);
SL(2,3)
gap> Order(G);
24
gap> L;
[ (2,4), (1,3,2,4), (1,4,2), (1,4,2), (1,4) ]
gap> x := L[1];
(2,4)
gap> Add(L,x);
gap> L;
[ (2,4), (1,3,2,4), (1,4,2), (1,4,2), (1,4), (2,4) ]
gap> S := Set(L);
[ (2,4), (1,3,2,4), (1,4,2), (1,4) ]
gap>
```

And so let us see what as of now, L is this and let me just say x is suppose the first element of L. So, L is this first element of the list and I add to this list the element x again, what happens.

So, you can see in the end it is 2 4 here, right, the 2 4 is here in end. So, there is a duplication in the list which set does not have duplication. So, I can convert this list. So, I can have some S which is set of L and now it is just one at a time. So, set is having one entry exactly once, but list can have multiple times same entry multi mystic could be there. So, that is S sometimes S S are useful sometimes keeping your data in the form of set is useful depends on the context. So, we have how we have, how to define groups in GAP you know how to manage lists and sets and looping also we have seen that we can (Refer Time: 22:44).

And now we are going to understand how one can put Rubik’s cube in the gap. So, let us see and after that we write a program to solve Rubik’s cube, right. So, that is a group theory problem for us. First let us see how Rubik’s cube can be stored inside gap. So, when I say Rubik’s cube what actually mean is group the Rubik’s group.

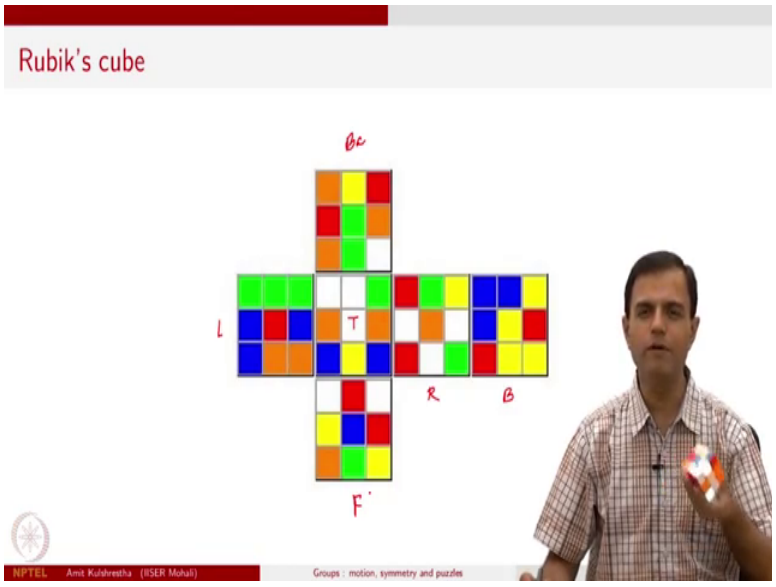
(Refer Slide Time: 23:26)

Rubik's cube

NPTEL Amit Kulshrestha (IISER Mohali)
Groups : motion, symmetry and puzzles

So, that is a Rubik's cube. What do I do? I just open this I just open this in the way that here I have say top, here I have say left, here I have right. So, here I have say bottom and this is back and this is say front.

(Refer Slide Time: 23:38)

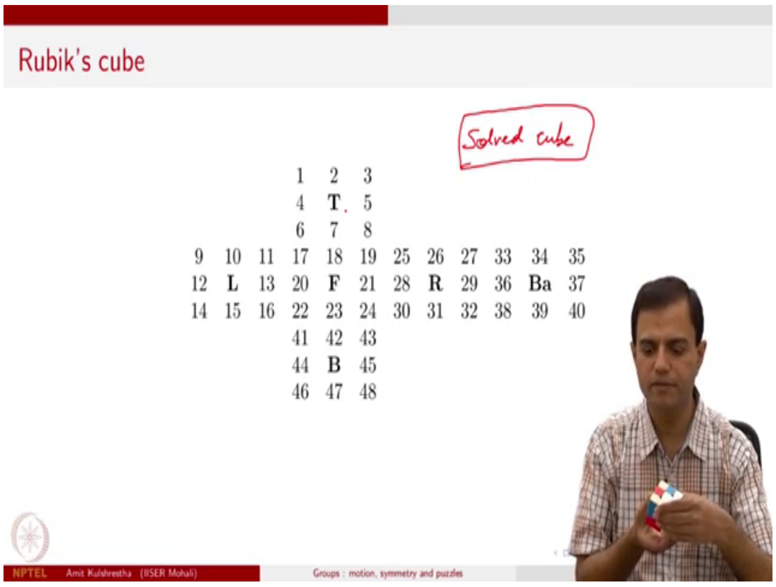


Rubik's cube

Bc
L
T
R
B
F
NPTEL Amit Kulshrestha (IISER Mohali)
Groups : motion, symmetry and puzzles

So, I can open my Rubik's cube in this fashion right, and everything is about permutation. So, let us see what kind of permutations happen.

(Refer Slide Time: 24:16)



Rubik's cube

Solved cube

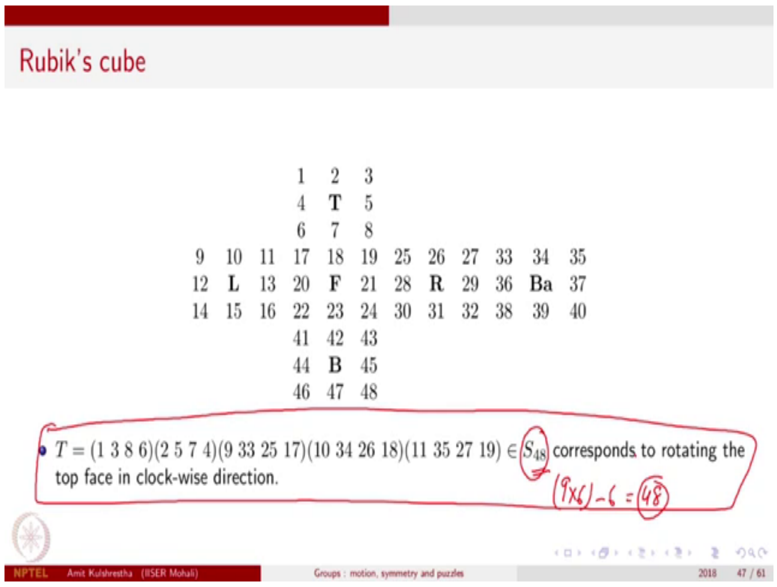
			1	2	3						
			4	**T**	5						
			6	7	8						
9	10	11	17	18	19	25	26	27	33	34	35
12	**L**	13	20	**F**	21	28	**R**	29	36	**Ba**	37
14	15	16	22	23	24	30	31	32	38	39	40
			41	42	43						
			44	**B**	45						
			46	47	48						

NPTEL Amit Kulshrestha (IISER Mohali) Groups : motion, symmetry and puzzles

So, for that I am going to present solved cube in this fashion. So, I am going to name all the this face all the small portions all the small squares of tiles, on the periphery, on the edge and I am going to name and this stands for t stands for top. So, I am just opening it like this. So, so the way I had put the nomenclature in the earlier stage is slightly different here I am opening it this way, this is, so that this is front then that is top that is left right back and the bottom down side.

So, I have named all on suppose, all the small small tiles by 1 2 3. So, let me just say that this is solved cube, solved Rubik’s cube. So, I have name 1 2 3 4 and top 5 6 7 8 and like that. And now it is interesting what happens when I rotate.

(Refer Slide Time: 25:23)



Rubik's cube

			1	2	3						
			4	**T**	5						
			6	7	8						
9	10	11	17	18	19	25	26	27	33	34	35
12	**L**	13	20	**F**	21	28	**R**	29	36	**Ba**	37
14	15	16	22	23	24	30	31	32	38	39	40
			41	42	43						
			44	**B**	45						
			46	47	48						

- $T = (1\ 3\ 8\ 6)(2\ 5\ 7\ 4)(9\ 33\ 25\ 17)(10\ 34\ 26\ 18)(11\ 35\ 27\ 19) \in S_{48}$ corresponds to rotating the top face in clock-wise direction.

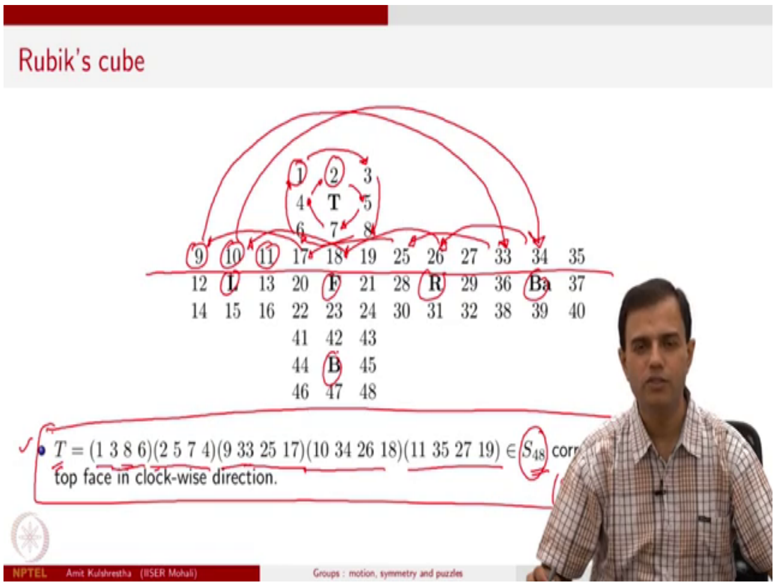
$(9\times 6)-6=48$

NPTEL Amit Kulshrestha (IISER Mohali) Groups : motion, symmetry and puzzles 2018 47 / 61

So, let us concentrate on this. First let us understand that the group which is relevant here is as 48. So, how does number 48 come, you have 9 small squares here and there are 6 faces 90 into 6 is 54, but as you see if I just do this, this one is not moving, the center one. As I move this the center one is at its place it is at its own place. So, it is not moving at all. So, there are 9 into 6 minus 6, only 48 numbers which are being permuted. So, 48 of these numbers which I written they are being permitted in the process. Let us go for the top thing.

So, what is happening when I am rotating by 90 degrees that is one single move, but one single move I am denoting by t and let us see what is happening when I am doing that what happens to 1, when I rotate by 90 degree.

(Refer Slide Time: 26:36)



Rubik's cube

- $T = (1\ 3\ 8\ 6)(2\ 5\ 7\ 4)(9\ 33\ 25\ 17)(10\ 34\ 26\ 18)(11\ 35\ 27\ 19) \in S_{48}$ corr top face in clock-wise direction.

NPTEL Amit Kulshrestha (IISER Mohali) Groups : motion, symmetry and puzzles

So, 1 comes to this place right. So, that is what I have written 1 goes to 3 and 3 goes to 8, 3 goes to 8, and then 8 goes to 6, 8 goes to 6 and 6 comes back to 1. So, one cycle is complete this is how we write permutations. As I had told you earlier, so 1 3 6 8, 1 3 6 8 is a permutation.

So, suppose it is a top one and I am rotating this is this is 8, but you have to be careful because when I am doing this there are 4 other sides. So, when I am moving the top one then except the bottom everything else is permuted right. So, this face also gets, this face also gets permutated and this front and back they also get permitted. So, in what fashion do they get permuted. Let us see what happens to 2. So, first let us just fix the top one. So, 2 goes to 5, 5 goes to 7, 7 goes to 4, 4 goes to 2. So, I have 2 5 7 4. So, top we have taken care of.

Now, what about the left one? So, all that is happening all the things that are happening are happening above this. So, when I moving top only the entries which are above this red line they are being affected bottom nothing happens. So, what happens to 9? So, 9 moves actually if you if you name it and if you label it play like this you would realize that 9 actually goes to 33. So, it goes like back. And then what happens to 33? 33 goes to 25, so 9 leave to 33 leave to 25, leave to 17 and then back to 9. So, this kind of cycles happens.

What happens to 10? Similar kind of thing happens to 10. So, 10 goes to 34, exactly the same fashion some translating fashion and goes to 34 and then 34 goes to leave to 26 and leave to 18 and 18 is again back to 10 and similarly you have 11.

So, all the permutations happen only at the top level, but except bottom every other face affects some permutations. So, this is a typical element and this represents rotating the top phase in clockwise direction. So, in this clockwise direction and moving the face, now, what can I do? I can actually write this kind of move for not just top, but for left front right back and bottom for all phases I can I can write these kind of moves. So, these are precisely what are generating Rubik's cube, right. So, Rubik's cube is actually generated by these kind of moves only, that is quite interesting.

How do find out how many such combinations are there? It is extremely difficult, because you know there are so many combinations there is so many combinations of this. So, let us see, let us see how we can store all these moves and then how can we process all that information into something which can solve Rubik's cube for us.

So, I hope you could understand how a single move is expressed in this. So, next time we are going to write all the moves of Rubik's cube and we are going to generate a group interesting thing would be to see how much the size of the group and then we would try to solve the cube if everything goes well. So, just wait and the meantime we want you can write other moves. So, other moves corresponding L, F, R, back and bottom, yeah. So, see you next time.