GAP

I. Introduction

You can start GAP on the PCs (from, chrystal, muir, shades) using the command gap-4. You can leave GAP using the command

```
gap>
```

```
quit;
```

GAP runs in a "Read", "Eval", "Print" (or rather "View", see later) loop. There is EMACS-like (or tcsh-like) command-line editing.

The help function can be called by "?Topic"

```
gap> ?group
```

Creating groups

The language is essentially unchanged from GAP3, it is Pascal-like.

Every object has a

```
Family and Filters (more
```

It is important to distinguish between equal and identical objects

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

Lists are also used to represent sets, strings, vectors, matrices:

```
gap> Reversed("Drab as a fool, aloof as a bard/");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```

Lists are also used to represent sets, strings, vectors, matrices:

```
Reversed("Drab as a fool, aloof as a bard!");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```

Lists are also used to represent sets, strings, vectors, matrices:

```
Reversed("Drab as a fool, aloof as abard!");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```

Lists are also used to represent sets, strings, vectors, matrices:

```
Reversed("Drab as a fool, aloof as a bard!");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```

Lists are also used to represent sets, strings, vectors, matrices:

```
Reversed("Drab as a fool, aloof as a bard!");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```

Lists are also used to represent sets, strings, vectors, matrices:

```
Reversed("Drab as a fool, aloof as a bard!");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```

Lists are also used to represent sets, strings, vectors, matrices:

```
Reversed("Drab as a fool, aloof as a bard!");
"drab a sa foola /,loof a sa barD/
```

```
List([i..m],i->i^2);
```

```
Size(g);
```

Sublists can be formed by using pointy brackets to indicate a range;

```
gap> [[1,2,3], [1,2,3,5]];
[ [ 1, 2, 3 ], [ 1, 2, 3, 5 ] ]
```

Lists are also used to represent sets, strings, vectors, matrices:

```
for var in list do
while condition do
next var until
repeat
if condition then
else
end;
```

```
[1,4,9,16,25,36,49,64,81,100]
```

```
Number([1..20],:isprime);
```

```
Filtered([1..1000],i->gcd(i,30)=1);
[ 6, 28, 496, 828 ]
```

The manual section "Operations for collections" contains more.

I. Introduction

```
 Alexander Hulpke, May 1, 1998
```
Finitely presented groups

GAP distinguishes between free groups and finitely presented groups:

gap> f:=FreeGroup("a","b");
<free group on the generators [ a, b ]>
gap> f:=FreeGroup("a","b","c");
<free group on the generators [ a, b, c ]>
gap> g:=f.f.1*f.2*f.3*f.2*f.1*f.3*f.1*f.2;
<fp group on the generators [ a, b, c ]>
gap> Size(g);
8
gap> g.1;
a
gap> f.1;
a
gap> g.1^f.1;
false
gap> g.1^g.1
true
gap> f.1^f.1^f.1;
false