SageMath Lecture 11 (Group Theory with SageMath)

Ajit Kumar @ ICT Mumbai, (INDAI)

June 24, 2020

1 Group Theory with SageMath

1.1 References

1. Abstract Algebra: Theory and Applications by By Tom Judson, Sage material by Rob Beezer
   A PDF and Sage worksheet versions of this tutorial are available at
   http://abstract.ups.edu/sage-aata.html
2. Group Theory: An expedition with SageMath by Ajit Kumar and Vikas Bist

1.2 Group of integers modulo n ($\mathbb{Z}/n\mathbb{Z}$ or $\mathbb{Z}_n$)

```
In [2]: n = 8
        G = Integers(n)
        G

Out[2]: Ring of integers modulo 8

In [8]: G.random_element()
        a = 78563726
        a in G

Out[8]: True

In [11]: G.list()
         g = G(672526)
         g

Out[11]: 6

In [12]: g.order()

Out[12]: 4

In [14]: g1 = G(634361)
         g1

Out[14]: 1

In [16]: g1*g
         g1+g
```

```
Out[16]: 7

In [18]: G.order()
         G.cardinality()

Out[18]: 8

In [19]: G.addition_table()

Out[19]: +  a b c d e f g h
          +----------------
         a| a b c d e f g h
         b| b c d e f g h a
         c| c d e f g h a b
         d| d e f g h a b c
         e| e f g h a b c d
         f| f g h a b c d e
         g| g h a b c d e f
         h| h a b c d e f g


In [20]: G.addition_table(names='elements')

Out[20]: +  0 1 2 3 4 5 6 7
          +----------------
         0| 0 1 2 3 4 5 6 7
         1| 1 2 3 4 5 6 7 0
         2| 2 3 4 5 6 7 0 1
         3| 3 4 5 6 7 0 1 2
         4| 4 5 6 7 0 1 2 3
         5| 5 6 7 0 1 2 3 4
         6| 6 7 0 1 2 3 4 5
         7| 7 0 1 2 3 4 5 6


In [22]: G.multiplication_table(names = 'elements')

Out[22]: *  0 1 2 3 4 5 6 7
          +----------------
         0| 0 0 0 0 0 0 0 0
         1| 0 1 2 3 4 5 6 7
         2| 0 2 4 6 0 2 4 6
         3| 0 3 6 1 4 7 2 5
         4| 0 4 0 4 0 4 0 4
         5| 0 5 2 7 4 1 6 3
         6| 0 6 4 2 0 6 4 2
         7| 0 7 6 5 4 3 2 1
```

1.3 Units in $\mathbb{Z}$ ($\mathbb{Z}_n^*$)

```
In [26]: n = 24
         G = Integers(n)
         U = [a for a in G if gcd(a,n)==1]
         U

Out[26]: [1, 5, 7, 11, 13, 17, 19, 23]

In [28]: H = G.unit_group()

In [30]: H.list()
         H.order()

Out[30]: 8

In [36]: (f0, f1, f2) = H.gens()
         f1.order()
         f2.order()
         (f1*f2).order()

Out[36]: 2
```

Exercise 1. Find the group of units of $\mathbb{Z}_{48}$.

1.4 Symmetric Group

```
In [37]: S4 = SymmetricGroup(4)
         S4

Out[37]: Symmetric group of order 4! as a permutation group

In [38]: S4.domain()

Out[38]: {1, 2, 3, 4}

In [39]: S4.order()

Out[39]: 24

In [40]: S4.list()

Out[40]: [(),
          (1,3)(2,4),
          (1,4)(2,3),
          (1,2)(3,4),
          (2,3,4),
          (1,3,2),
          (1,4,3),
          (1,2,4),
          (2,4,3),
          (1,3,4),
          (1,4,2),
          (1,2,3),
          (3,4),
          (1,3,2,4),
          (1,4,2,3),
          (1,2),
          (2,3),
          (1,3,4,2),
          (1,4),
          (1,2,4,3),
          (2,4),
          (1,3),
          (1,4,3,2),
          (1,2,3,4)]

In [41]: S4.is_abelian()

Out[41]: False

In [43]: g1 = S4.random_element()
         g1

Out[43]: (1,4)

In [44]: g2 = S4.random_element()
         g2

Out[44]: (1,4,2)

In [46]: g1*g2==g2*g1

Out[46]: False

In [47]: g2*g1

Out[47]: (2,4)

In [48]: S4.is_abelian()

Out[48]: False

In [51]: a = S4("(1,2,4)")
         a in S4

Out[51]: True

In [52]: b = S4("(1,3)")

In [54]: a.inverse()
```

```
Out[54]: (1,4,2)

In [55]: a.sign()

Out[55]: 1

In [57]: a1 = S4([4,3,1,2])
         a1 in S4

Out[57]: True

In [58]: a

Out[58]: (1,2,4)

In [59]: [a(x) for x in S4.domain()]

Out[59]: [2, 4, 3, 1]

In [60]: a.matrix()

Out[60]: [0 1 0 0]
         [0 0 0 1]
         [0 0 1 0]
         [1 0 0 0]

In [61]: S = SymmetricGroup(['A','B','C','D','E'])

In [64]: g1 = S.random_element()

In [67]: g1.parent()

Out[67]: Symmetric group of order 5! as a permutation group
```

1.5 Cycle as a product of transpositions

```
In [74]: G = SymmetricGroup(10)
         g1 = G.random_element()
         print(g1)

(1,8,9,10,5,3,6)(2,7,4)


In [76]: g = G('(1,6,2,4)(3,5)')
         g.cycle_type()

Out[76]: [4, 2, 1, 1, 1, 1]

In [72]: g1.cycle_type()

Out[72]: [10]
```

```
In [77]: def transpositions(rho):
             T=[]
             H = rho.parent()
             rho1=rho.cycle_tuples()[0]
             k = len(rho1)
             for i in range(k-1,0,-1):
                 i1=rho1[0]
                 i2=rho1[i]
                 r = H((i1,i2))
                 T.append(r)
             return(T)
```

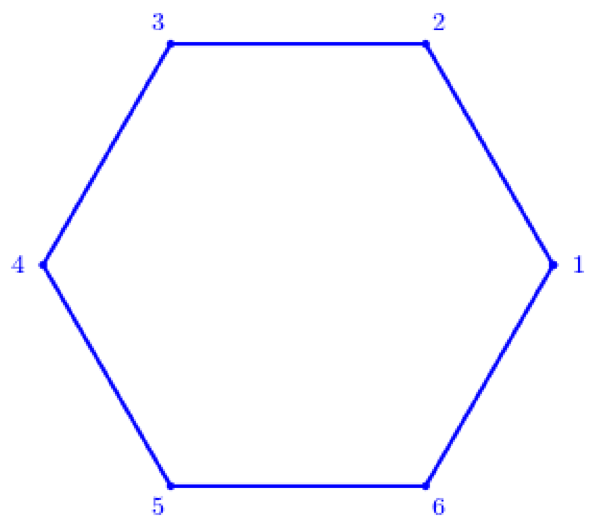
```
In [78]: G = SymmetricGroup(6)
         g1 = G("(1,3,2,4,5,6)")
```

```
In [82]: prod(tr)
```

```
Out[82]: (1,3,2,4,5,6)
```

1.6 Dihedral Group

```
In [1]: n = 6
        pts = [vector([cos(2*pi*i/n),sin(2*pi*i/n)]) for i in range(n+1)]
        vertices = point2d(pts,axes=False)
        ngon = line2d(pts,thickness=1.5)
        label = sum([text('$%s$'%(i+1), 1.1*pts[i]) for i in range(n)])
        show(vertices+ngon+label,aspect_ratio=1,figsize=4,)
```



```
In [2]: G = DihedralGroup(6)
        G
```

```
Out[2]: Dihedral group of order 12 as a permutation group

In [3]: G.list()

Out[3]: [(),
         (1,5,3)(2,6,4),
         (1,3,5)(2,4,6),
         (1,6,5,4,3,2),
         (1,4)(2,5)(3,6),
         (1,2,3,4,5,6),
         (2,6)(3,5),
         (1,5)(2,4),
         (1,3)(4,6),
         (1,6)(2,5)(3,4),
         (1,4)(2,3)(5,6),
         (1,2)(3,6)(4,5)]

In [4]: G.is_abelian()

Out[4]: False

In [5]: r = G('(1,3,5)(2,4,6)')

In [6]: r.order()

Out[6]: 3

In [7]: [a.order() for a in G]

Out[7]: [1, 3, 3, 6, 2, 6, 2, 2, 2, 2, 2, 2]
```

1.7 Alternating Group

```
In [8]: A5 = AlternatingGroup(5)

In [9]: A5.is_simple()

Out[9]: True

In [10]: A5.gens()

Out[10]: [(3,4,5), (1,2,3,4,5)]

In [11]: A5.one()
         A5.identity()

Out[11]: ()

In [12]: a = A5.random_element()
         a.inverse()*a==A5.one()

Out[12]: True
```

1.8 Quaternion and Klien Four

```
In [13]: # Klien Four Group
         K4 = KleinFourGroup()
         K4.list()

Out[13]: [(), (3,4), (1,2), (1,2)(3,4)]

In [14]: ## Quaternion Group
         Q = QuaternionGroup()
         Q.order()

Out[14]: 8

In [15]: Q.list()
         Q.cayley_table()

Out[15]: *  a b c d e f g h
          +----------------
         a| a b c d e f g h
         b| b c d a h e f g
         c| c d a b g h e f
         d| d a b c f g h e
         e| e f g h c d a b
         f| f g h e b c d a
         g| g h e f a b c d
         h| h e f g d a b c


In [ ]:
```

1.9 Matrix Group

```
In [ ]: groups.matrix.

In [17]: G = GL(3,RR) # set of all 3 x 3 real invertible matrices
         G

Out[17]: General Linear Group of degree 3 over Real Field with 53 bits of precision

In [18]: G.is_finite()

Out[18]: False

In [19]: G.cardinality()

Out[19]: +Infinity

In [21]: SL(3,ZZ) # set of all 3 x 3 matrices of determinant 1 over integer

Out[21]: Special Linear Group of degree 3 over Integer Ring
```

```
In [22]: GL3Z = GL(3,ZZ)
         gens = GL3Z.gens()
         A = gens[1]
         A.order()
         gens

Out[22]: (
         [0 1 0]  [0 1 0]  [-1  0  0]  [1 1 0]
         [0 0 1]  [1 0 0]  [ 0  1  0]  [0 1 0]
         [1 0 0], [0 0 1], [ 0  0  1], [0 0 1]
         )

In [23]: m1 = gens[1]
         m2 = gens[2]
         H = MatrixGroup([m1,m2])
         H.order()

Out[23]: 8

In [24]: H.structure_description()

Out[24]: 'D4'

In [120]: H.is_isomorphic(DihedralGroup(4))

Out[120]: True
```

1.9.1 Matrix Groups over Finite Field

```
In [25]: p = 5
         F = GF(p)

In [26]: G = GL(3,F)
         G

Out[26]: General Linear Group of degree 3 over Finite Field of size 5

In [27]: G.is_finite()

Out[27]: True

In [28]: G.order()

Out[28]: 1488000
```

1.9.2 Order of GL(n, F(p))

No of elements in the 1st row $p^n - 1$
  No. of elements in the second row: (not multiple of he 1st row) $p^n - p$
  No. of elements in the 3rd row: $p^n - p^2 \dots$
  No. of elements in the n-th row: $p^n - p^{n-1}$
  Total = $(p^n - 1)(p^n - p)(p^n - p^2) \cdots (p^n - p^{n-1})$.

```
In [29]: G.order()

Out[29]: 1488000

In [30]: (p^3-1)*(p^3-p)*(p^3-p^2)

Out[30]: 1488000

In [31]: G = SL(3,F)
         G.order()

Out[31]: 372000

In [32]: (p^3-1)*(p^3-p)*(p^3-p^2)/(p-1)

Out[32]: 372000
```

1.9.3 Order of $SL_n(F(p))$

Consider a homomorphism $\varphi \colon GL_n(F(p)) \to F(p)^*$ given by $A \to \det(A)$. Then the kernal of φ is $SL_n(F(p))$. Thus

$$|GL_n(F(p))| = (p-1) \times |SL_n(F(p))|,$$

In particular

$$|SL_n(F(p))| = \frac{|GL_n(F(p))|}{(p-1)}.$$

```
In [33]: G = GL(2,Zmod(4))
         print(G.cardinality())
         G.structure_description()

96


Out[33]: '(C2 x S4) : C2'
```

1.9.4 Quaternion Group as a matrix group.

```
In [35]: G = groups.matrix.QuaternionGF3()
         G.list()

Out[35]: (
         [0 1]  [0 2]  [1 0]  [1 1]  [1 2]  [2 0]  [2 1]  [2 2]
         [2 0], [1 0], [0 1], [1 2], [2 2], [0 2], [1 1], [2 1]
         )
```

Exercise
Create a matrix group with $M1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ and $M2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ as generators over $\mathbb{Z}$.
Find the order of this group and list all its elements.

```
In [ ]: M1 = matrix(ZZ,2,[[-1,0],[0,1]])
        M2 = matrix(ZZ,2,[[1,0],[0,-1]])
        G = MatrixGroup([M1, M2])
        G.list()
        SLn(F(p
```

Exer Consider the group $GL_4(\mathbb{F}_2)$. Check that it is isomorphic to $\mathbb{A}_8$.

```
In [ ]: G = GL(4,2)
        G.order()
```

```
In [ ]: G.is_isomorphic(AlternatingGroup(8))
```

```
In [ ]: G.structure_description()
```

```
In [ ]: M1 = matrix(GF(11), [[1,2],[3,4]])
        M2 = matrix(GF(11), [[1,0],[5,1]])
        G = MatrixGroup([M1,M2])
        G.cardinality()
        G.structure_description()
```

1.9.5 Intersection of two groups

```
In [41]: A = Matrix([(0, 1/2, 0), (2, 0, 0), (0, 0, 1)])
         B = Matrix([(0, 1/2, 0), (-2, -1, 2), (0, 0, 1)])
         G = MatrixGroup([A,B])
         U = G.intersection(GL(3,ZZ))
         V = G.intersection(SL(3,ZZ))
         show(U)
         show(V)
```

```
Subgroup with 1 generators (
[ 1  0  0]
[-2 -1  2]
[ 0  0  1]
) of Matrix group over Rational Field with 2 generators (
[   0 1/2   0]  [   0 1/2   0]
[   2   0   0]  [  -2  -1   2]
[   0   0   1], [   0   0   1]
)


Subgroup with 0 generators () of Matrix group over Rational Field with 2 generators (
[   0 1/2   0]  [   0 1/2   0]
[   2   0   0]  [  -2  -1   2]
[  0   0   1], [  0   0   1]
)
```

```
In [ ]:
```

1.10 Playing with orders

```
In [138]: G = SymmetricGroup(10)
          x = G.random_element()
          print(x)

(1,6,4,5,3,2)(7,8,9,10)


In [139]: x = G('(1,3,6,7)')
          y = G('(2,4,5,8,9,10)')

In [140]: x*y==y*x

Out[140]: True

In [141]: (x*y).order()

Out[141]: 12

In [143]: a = G.random_element()
          a.order() == (~a).order()

Out[143]: True
```

1.11 Exponent of a group

```
In [42]: G = SymmetricGroup(6)
         G.exponent()

Out[42]: 60

In [43]: lcm([a.order() for a in G])

Out[43]: 60

In [44]: for i in range(1, G.order()+1):
             idx = 0
             index = i
             for a in G:
                 if(a^i==G.identity()):
                     idx=idx+1

             if(idx==G.order()):
                 index = i
                 break

         index

Out[44]: 60
```

1.12 Subgroups

```
In [45]: G = DihedralGroup(6)
         subgrps = G.subgroups()
         len(subgrps)

Out[45]: 16

In [46]: G = SymmetricGroup(4)
         g1 = G("(1,2,4)")
         g2 = G("(1,4)")
         H1 = G.subgroup([g1])

         H.order()
         H2 = G.subgroup([g1,g2])

In [153]: H2.is_subgroup(G)

Out[153]: True
```

1.13 Cosets

```
In [47]: G =  SymmetricGroup(4)
         g1 = G("(1,2,4)")
         g2 = G("(1,4)")
         H = G.subgroup([g1,g2])

In [48]: H.order()

Out[48]: 6

In [49]: rc = G.cosets(H,side='right')
         rc

Out[49]: [[(), (2,4), (1,2), (1,2,4), (1,4,2), (1,4)],
          [(3,4), (2,3,4), (1,2)(3,4), (1,2,3,4), (1,3,4,2), (1,3,4)],
          [(2,3), (2,4,3), (1,3,2), (1,3,2,4), (1,4,3,2), (1,4)(2,3)],
          [(1,2,3), (1,2,4,3), (1,3), (1,3)(2,4), (1,4,3), (1,4,2,3)]]

In [50]: lc = G.cosets(H,side='left')
         lc

Out[50]: [[(), (2,4), (1,2), (1,2,4), (1,4,2), (1,4)],
          [(3,4), (2,4,3), (1,2)(3,4), (1,2,4,3), (1,4,3,2), (1,4,3)],
          [(2,3), (2,3,4), (1,2,3), (1,2,3,4), (1,4,2,3), (1,4)(2,3)],
          [(1,3,2), (1,3,4,2), (1,3), (1,3,4), (1,3)(2,4), (1,3,2,4)]]

In [51]: lc == rc

Out[51]: False
```

```
In [52]: rc_sorted = sorted([sorted(C) for C in rc])
         lc_sorted = sorted([sorted(C) for C in lc])

In [53]: rc_sorted==lc_sorted

Out[53]: False

In [54]: N = G.normal_subgroups()
         N

Out[54]: [Subgroup generated by [(1,2), (1,2,3,4)] of (Symmetric group of order 4! as a permutat
          Subgroup generated by [(2,3,4), (1,2,3)] of (Symmetric group of order 4! as a permutat
          Subgroup generated by [(1,2)(3,4), (1,4)(2,3)] of (Symmetric group of order 4! as a pe
          Subgroup generated by [()] of (Symmetric group of order 4! as a permutation group)]

In [55]: [H.order() for H in N]

Out[55]: [24, 12, 4, 1]

In [56]: H1 = G.subgroup([G('(1,3)(2,4)'), G("(1,4)(2,3)")])

In [57]: H1.order()
         H1.is_normal(G)

Out[57]: True

In [58]: rc = G.cosets(H1,side='right')
         lc = G.cosets(H1,side='left')
         lc==rc

Out[58]: False

In [59]: rc_sorted = sorted([sorted(C) for C in rc])
         lc_sorted = sorted([sorted(C) for C in lc])
         lc_sorted==rc_sorted

Out[59]: True

In [ ]:
```

SageMath: Lecture 12
Group Theory with SageMath

Ajit Kumar
ICT Mumbai (INDIA)

June 29, 2020

1 SageMath Lecturre 12

1.1 Subgroups

Example Let us explore the permutation group generated by (3,4)(6,7) and (1,4,7).

$$G = \langle\{(1,4,7),(3,4)(6,7)\}\rangle.$$

```
In [1]: ## Defining the group
        G = PermutationGroup([[(1,4,7)],[(4,3),(6,7)]])
```

```
In [2]: ## Generators of G
        G.gens()
```

```
Out[2]: [(3,4)(6,7), (1,4,7)]
```

```
In [3]: ## Order of the group
        G.order()
```

```
Out[3]: 60
```

```
In [4]: ## Random element in G and its order
        g1 = G.random_element()
        print(g1)
        print(g1.order())
```

```
(1,3,6)
3
```

```
In [5]: ## set of order of all elements of G (what do you observe?)
        set([a.order() for a in G])
```

```
Out[5]: {1, 2, 3, 5}
```

```
In [6]: ## All (number of ) subgroups of G
        subgrps = G.subgroups()
        len(subgrps)
```

```
Out[6]: 59

In [7]: ## All (number of ) normal subgroups of G (Is G simple?)
        normalsubgrps = G.normal_subgroups()
        len(normalsubgrps)

Out[7]: 2

In [8]: ## Set of orders of subgroups of G
        ## Does it have subgroup of every division of the order of G?
        set([H.order() for H in subgrps])

Out[8]: {1, 2, 3, 4, 5, 6, 10, 12, 60}

In [9]: ## Number of subgroups of order 6
        subgrgp6 = [H for H in subgrps if(H.order()==6)]
        len(subgrgp6)

Out[9]: 10

In [10]: ## Number of subgroups of order 4
         subgrgp4= [H for H in subgrps if(H.order()==4)]
         len(subgrgp4)

Out[10]: 5

In [11]: ## Conder two subgroup H and K of G and find their product KH.
         ## Is HK a subgroup of G?
         H = subgrgp6[1]
         K = subgrgp6[2]

In [12]: ## Intersection of H and K
         U = H.intersection(K);U
         U.order()

Out[12]: 2

In [13]: ## The product of HK
         HK = set([h*k for h in H for k in K])
         len(HK)

Out[13]: 18

In [14]: ## The product of KH
         KH = set([k*h for h in H for k in K])
         len(KH)

Out[14]: 18

In [15]: ## Is HK=KH?
         sorted(KH)==sorted(HK)
```

```
Out[15]: False

In [16]: ## IS |HK| =|H||K|/|H intersection K|
         H.order()*K.order()/U.order()

Out[16]: 18
```

Example:
Consider the group $\mathbb{S}_4$. Find subgroups H and K such that HK is not a subgroup.

```
In [17]: G = SymmetricGroup(4)
         H = G.subgroup([G("(1,2,3)")])
         K = G.subgroup([G("(1,4)(2,3)")])

In [18]: print(H.order(),K.order())

3 2


In [19]: HK = set(flatten([[x*y for x in H] for y in K]))
         len(HK)

Out[19]: 6

In [20]: for a in HK:
             for b in HK:
                 if(a*b not in HK):
                     break
         print(a,b)

(1,2,4) (1,2,3)


In [21]: a*b in HK

Out[21]: False
```

1.2 Cyclic Permutation Groups

```
In [22]: n = 30
         H = CyclicPermutationGroup(n);

In [23]: len(H.subgroups())
         len(H.normal_subgroups())

Out[23]: 8
```

Exerercise
Find all subgroups of C_{30}.

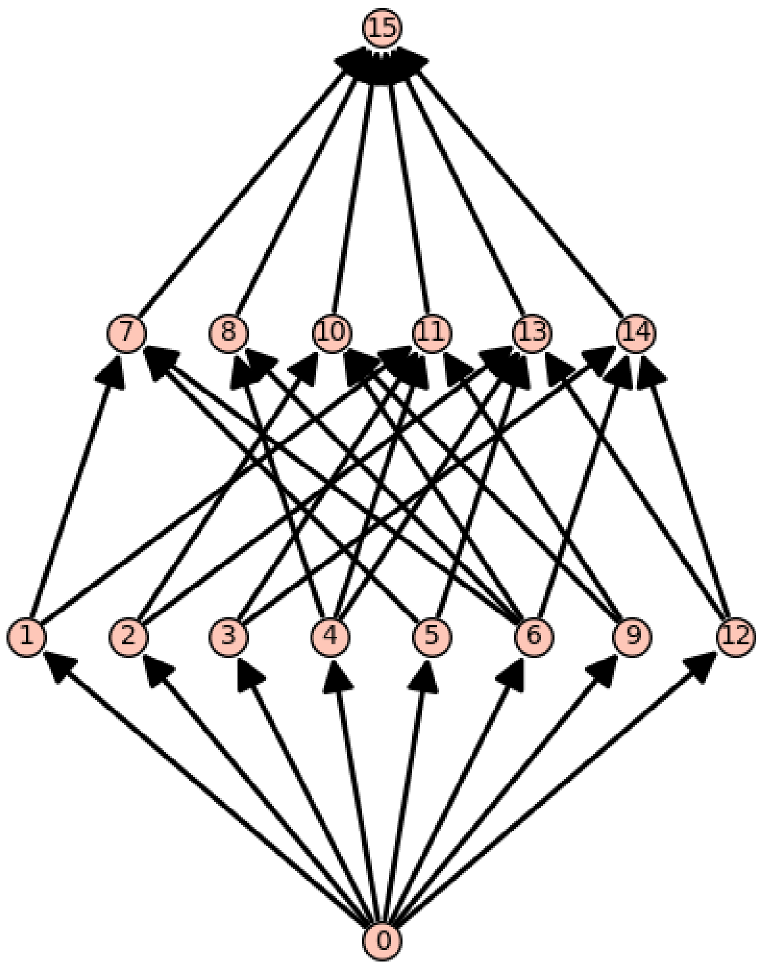
1.3 Multiplicative Abelian Groups and Free Groups

```
In [24]: G = AbelianGroup([8])
         G.gens()

Out[24]: (f,)

In [25]: ## Z as abelian group
         AbelianGroup([0])

Out[25]: Multiplicative Abelian group isomorphic to Z

In [26]: # Group C3 x C5
         G = AbelianGroup([3,5])
         G

Out[26]: Multiplicative Abelian group isomorphic to C3 x C5

In [27]: ## C3 x C5 as permutation group
         G.permutation_group()

Out[27]: Permutation Group with generators [(4,5,6,7,8), (1,2,3)]

In [28]: # The group C6 x C8 with generators a and b
         G.<a,b> = AbelianGroup([6,8])
         G.gens()

Out[28]: (a, b)

In [29]: ## Subgroups of C2 x C4 x C6
         len(AbelianGroup([2,4,6]).subgroups())

Out[29]: 54

In [30]: ## Free groups on {a,b}
         G = FreeGroup([a,b])
         G

Out[30]: Free Group on generators {a, b}

In [33]: ## Free group on {x0,x1,x2}
         G = FreeGroup(3,'x')
         G

Out[33]: Free Group on generators {x0, x1, x2}

In [35]: y = G([2, -3, 2, 1, 3, -1, -2])
         y

Out[35]: x1*x2^-1*x1*x0*x2*x0^-1*x1^-1
```

1.4 Lattice (Hasse Diagram) of subgroups

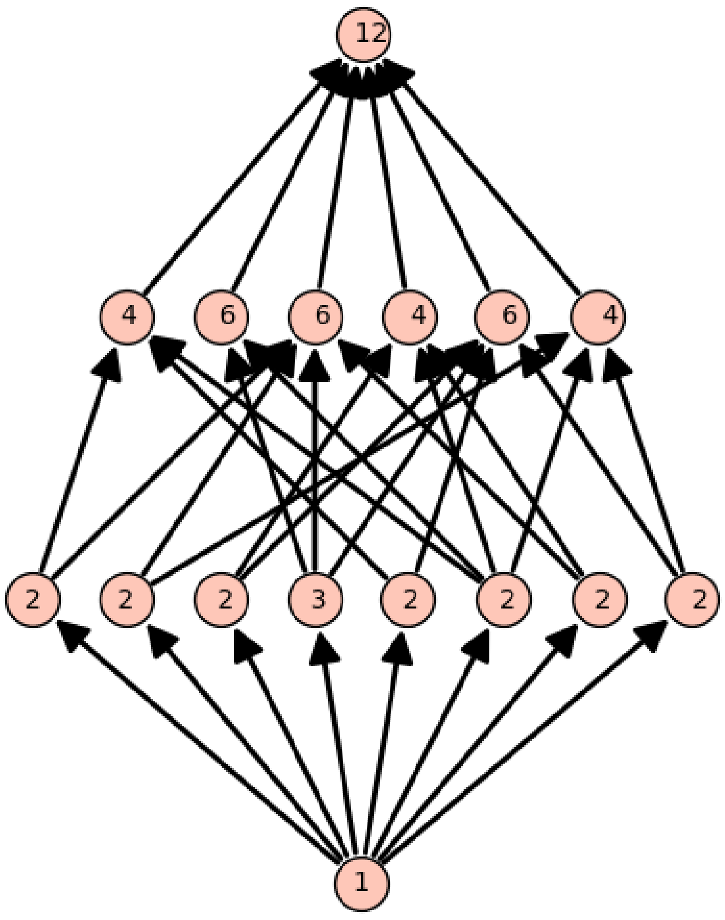
```
In [39]: G = DihedralGroup(6)
         subgroups = G.subgroups()
         P = Poset((subgroups, lambda h,k: h.is_subgroup(k)))
         relabeling = {h:i for (i,h) in enumerate(P)}
         Q = P.relabel(relabeling)
         Q.plot(figsize=8)
```

```
Out[39]:
```



```
In [38]: G = DihedralGroup(6)
         subgroups = G.subgroups()
         P = Poset((subgroups, lambda h,k: h.is_subgroup(k)))
         S = [" "*floor(i/2)+str(len(P[i]))+" "*floor(i/3)  for i in range(len(P))]
         label = {P[i]: S[i] for i in range(len(P))}
         Q = P.relabel(label)
```

```
D6_L=Q.plot(figsize=8,vertex_size=400,title=r"Lattice of dihedral group $D_6$",
                    title_pos=(0.5,-0.1),fontsize=12)
show(D6_L,figsize=8)
```



Lattice of dihedral group D_6

1.5 Group Homomorphism in Sagemth

SageMath can create group homomorphism between finite permutation groups.

The SageMath syntax "PermutationGroupMorphism()" is used to construct homomorphism between two groups.

It requires two groups and a list of images of generators of the domain group.

Example Let us set-up a homomorphism φ from the cyclic group C_{12} to D_8. If g is a generator of C_{12}, then we map g to x^2 where x rotation in D_8.

```
In [40]: G = CyclicPermutationGroup (12)
         H= DihedralGroup(8)
```

```
In [41]: x = G.gens()[0]
         y = H.gens()[0]

In [42]: phi = PermutationGroupMorphism(G,H,[y^2])

In [43]: K = phi.kernel()
         print(K.order())
         K.is_normal(G)

3


Out[43]: True

In [44]: a, b = G.random_element(), G.random_element()
         phi(G.one())

Out[44]: ()

In [45]: Img = phi.image(G)
         Img.order()

Out[45]: 4

In [46]: Img.is_subgroup(H)

Out[46]: True

In [47]: Q = G.quotient(K)
         Q.is_isomorphic(Img)

Out[47]: True
```

1.6 Group Action

Action of S_4 on $\mathbb{Q}^4$ as

$$\sigma \cdot (x_1, x_2, x_3, x_4) = (x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)}, x_{\sigma(4)}).$$

```
In [53]: G = SymmetricGroup(4)
         sig = G.random_element();
         print(sig)

(2,3,4)


In [54]: sig.matrix().T

Out[54]: [1 0 0 0]
         [0 0 0 1]
         [0 1 0 0]
         [0 0 1 0]
```

```
In [55]: def action2(sig,v):
             M=sig.matrix().T
             return(M*v)

In [56]: G = SymmetricGroup(4)
         X =QQ^4
         v = X.random_element();
         print(v)
         sig = G.random_element();
         print(sig)
         action2(sig,v)

(0, 0, -1, 0)
(1,4,2,3)


Out[56]: (-1, 0, 0, 0)

In [57]: v = X.random_element()
         action2(G.one(),v)==v

Out[57]: True

In [58]: sig,tau = G.random_element(),G.random_element()
         print(sig,tau)
         action2(sig*tau,v)==action2(tau, action2(sig,v))

(1,2,4,3) (1,4,2,3)


Out[58]: True
```

1.7 Conjugacy Action

```
In [59]: G = PermutationGroup([[(1,4,7)],[(4,3),(6,7)]])

In [60]: ## G.orbits() Returns the orbits of the elements of the domain under the default group
         G.orbits() # Action of G on to vertices {1,2,3,4,5,6,7}

Out[60]: [[1, 4, 3, 7, 6], [2], [5]]

In [61]: ## Centerlizer of an element x in X
         x = G("(1,6)(3,4)")
         Cg = G.centralizer(x)
         list(Cg)
         Cg.is_subgroup(G)

Out[61]: True
```

```
In [62]: ## Normalizer of an element g in G
         g = G.random_element()
         Ng = G.normalizer(g)
         print(Ng)

Subgroup generated by [(3,7)(4,6), (1,4,6)] of (Permutation Group with generators [(3,4)(6,7), (
```

```
In [63]: ## Stabiizer of the vertex 4
         Stab4 = G.stabilizer(4)
         Stab4.is_subgroup(G)

Out[63]: True
```

```
In [64]: ## Conjugacy class of g
         CGg = G.conjugacy_class(g)
         CGg

Out[64]: Conjugacy class of (1,4,6) in Permutation Group with generators [(3,4)(6,7), (1,4,7)]
```

```
In [65]: ## All conjugacy classes
         G.conjugacy_classes()

Out[65]: [Conjugacy class of () in Permutation Group with generators [(3,4)(6,7), (1,4,7)],
          Conjugacy class of (4,6,7) in Permutation Group with generators [(3,4)(6,7), (1,4,7)],
          Conjugacy class of (3,4)(6,7) in Permutation Group with generators [(3,4)(6,7), (1,4,7
          Conjugacy class of (1,3,4,6,7) in Permutation Group with generators [(3,4)(6,7), (1,4,
          Conjugacy class of (1,3,4,7,6) in Permutation Group with generators [(3,4)(6,7), (1,4,
```

```
In [66]: ## Conjugacy class representatives
         G.conjugacy_classes_representatives()

Out[66]: [(), (4,6,7), (3,4)(6,7), (1,3,4,6,7), (1,3,4,7,6)]
```

1.8 Class Equation

$$|X| = |X^G| + \sum_{x_i \notin X^G} [G : \mathsf{Stab}_G(x_i)]. \tag{1}$$

The above formula is called the **class equation.**

```
In [75]: CCR = G.conjugacy_classes_representatives()
         NX = [G.order()/G.centralizer(g).order() for g in CCR]
         sum(NX)==G.cardinality()

Out[75]: True
```

1.9 Sylow subgroups

```
In [67]: ## A Sylow 2 subgroup in G
         Sp2=G.sylow_subgroup(2)
         Sp2

Out[67]: Subgroup generated by [(3,4)(6,7), (3,6)(4,7)] of (Permutation Group with generators [(

In [68]: ## A Sylow 3 subgroup in G
         Sp3=G.sylow_subgroup(3)
         Sp2

Out[68]: Subgroup generated by [(3,4)(6,7), (3,6)(4,7)] of (Permutation Group with generators [(

In [69]: ## A Sylow 5 subgroup in G
         Sp5=G.sylow_subgroup(5)
         Sp5

Out[69]: Subgroup generated by [(1,7,4,3,6)] of (Permutation Group with generators [(3,4)(6,7),

In [70]: ## Number of Sylow Subgroups
         def Sylow_pgrous(G, p):
             SSGp = []
             P = G.sylow_subgroup(p)
             for g in G:
                 H = P.conjugate(g)
                 if not(H in SSGp):
                     SSGp.append(H)
             return SSGp

In [71]: Sylow_pgrous(G,2)

Out[71]: [Permutation Group with generators [(3,4)(6,7), (3,7)(4,6)],
          Permutation Group with generators [(1,3)(4,6), (1,6)(3,4)],
          Permutation Group with generators [(1,4)(3,7), (1,7)(3,4)],
          Permutation Group with generators [(1,3)(6,7), (1,6)(3,7)],
          Permutation Group with generators [(1,4)(6,7), (1,7)(4,6)]]

In [72]: Sylow_pgrous(G,3)

Out[72]: [Permutation Group with generators [(4,6,7)],
          Permutation Group with generators [(1,6,4)],
          Permutation Group with generators [(1,7,3)],
          Permutation Group with generators [(1,7,6)],
          Permutation Group with generators [(3,7,6)],
          Permutation Group with generators [(3,4,6)],
          Permutation Group with generators [(3,7,4)],
          Permutation Group with generators [(1,6,3)],
          Permutation Group with generators [(1,7,4)],
          Permutation Group with generators [(1,4,3)]]
```

```
In [73]: Sylow_pgrous(G,5)

Out[73]: [Permutation Group with generators [(1,4,3,7,6)],
          Permutation Group with generators [(1,3,4,6,7)],
          Permutation Group with generators [(1,6,4,7,3)],
          Permutation Group with generators [(1,7,3,6,4)],
          Permutation Group with generators [(1,6,3,4,7)],
          Permutation Group with generators [(1,4,7,6,3)]]

In [ ]:
```

1.10 Example

Let p be a prime. Define

$$A_p := \left\{ \begin{pmatrix} a & b \\ 0 & 1 \end{pmatrix} : a, b \in \mathbb{Z}_{p^2}, a \equiv 1 \pmod{p} \right\}.$$

```
In [76]: p = 5
         D = Zmod(p^2)

         D1 = [a for a in  D if((a-1)%p==0)]
         D1
         L = [[a,b,0,1] for a in D1 for b in D]
         G = MatrixGroup([matrix(2,2, l) for l in L])
         G.cardinality()

Out[76]: 125
```

Example
Find a Sylow 2 subgroup in $G = SL(2,5)$.

```
In [77]: ## Example: G = GL(2,5). Find a sylow-p groups in $G$.
         n,p=2,5
         G = GL(n,GF(p))
         print(G.order())
         print(factor(G.order()))

480
2^5 * 3 * 5


In [78]: set([A.order() for A in G])

Out[78]: {1, 2, 3, 4, 5, 6, 8, 10, 12, 20, 24}

In [79]: O2 = [g for g in G if g.multiplicative_order()==2]
         O4 = [g for g in G if g.multiplicative_order()==4]
         (len(O2),len(O4))
```

```
Out[79]: (31, 152)
```

```
In [80]: for i in range(len(O2)):
             for j in range(len(O4)):
                 A,B= O2[i],O4[j]
                 H = MatrixGroup([matrix(GF(p),n,flatten(A.list())),
                                       matrix(GF(p),n,flatten(B.list()))]);
                 if(H.order()==32):
                     break
         print(H.order())

32
```

```
In [81]: H

Out[81]: Matrix group over Finite Field of size 5 with 2 generators (
         [4 0]  [3 3]
         [1 1], [1 0]
         )
```

1.11 Group of Symmetries

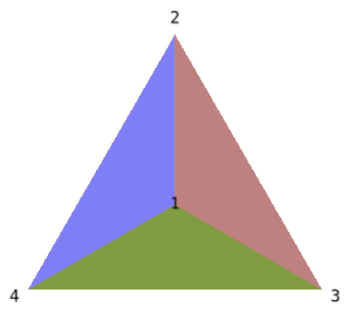
1.11.1 Group of symmetries of a tetrahedron

Demo on axis of rotation of tetrahedron
https://vimeo.com/90519331

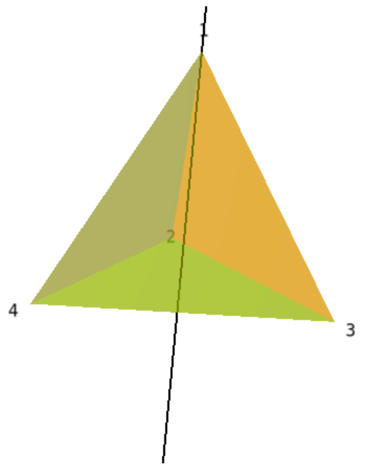
```
In [131]: #tetrahedron(frame=False)
```

```
In [83]: ## Plotting Tetrahedron
         v1 = vector([1, 1, 1])
         v2 = vector([-1, -1, 1])
         v3 = vector([-1, 1, -1])
         v4 = vector([1, -1, -1])
         G1 = (v2+v3+v4)/3
         G2 = (v1+v3+v4)/3
         G3 = (v2+v1+v4)/3
         G4 = (v2+v3+v1)/3
         s12 = (v1+v2)/2
         s34 = (v3+v4)/2
         s23 = (v2+v3)/2
         s24 = (v2+v4)/2
         s14 = (v1+v4)/2
         s13 = (v1+v3)/2
         tetrahd = polygon3d([v1,v2,v3],color='red',opacity=0.5)\
             +polygon3d([v1,v2,v4],color='blue',opacity=0.5)+\
                 polygon3d([v2,v3,v4],color='green',opacity=0.5)+\
                 polygon3d([v1,v3,v4],color='yellow',opacity=0.5)
```

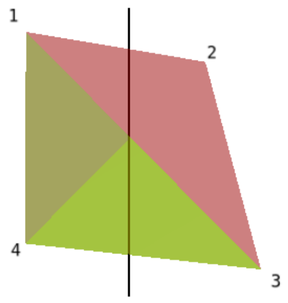
```
t1 = text3d('1',1.1*v1)
t2 = text3d('2',1.1*v2)
t3 = text3d('3',1.1*v3)
t4 = text3d('4',1.1*v4)
txt = t1+t2+t3+t4
tetrahd = tetrahd+txt
show(tetrahd,frame=False,figsize=3)
```



```
In [84]: var('t')
         v1L = parametric_plot3d(t*v1+(1-t)*(G1-v1),(t,-0.1,1.1),\
                                 linestyle="--",color='black')
         (v1L+tetrahd).show(frame=False)
```



```
In [85]: var('t')
         ML = parametric_plot3d(t*s12+(1-t)*s34,(t,-0.2,1.2),\
                                  linestyle="--",color='black')
         (ML+tetrahd).show(frame=False)
```



```
In [87]: show(tetrahd,frame=False)

In [88]: tetrahedron1 = PermutationGroup(["(1,2,4)", "(2,3,4)"])

In [89]: tetrahedron1.order()
```

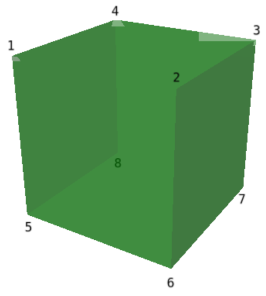
```
Out[89]: 12

In [90]: [a.order() for a in tetrahedron1]

Out[90]: [1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3]

In [91]: subgrps = tetrahedron1.subgroups()

In [92]: [K.order() for K in subgrps]

Out[92]: [1, 2, 2, 2, 3, 3, 3, 3, 4, 12]

In [93]: tetrahedron1.is_isomorphic(DihedralGroup(6))

Out[93]: False

In [94]: tetrahedron1.is_isomorphic(AlternatingGroup(4))

Out[94]: True

In [95]: tetrahedron1.structure_description()

Out[95]: 'A4'
```

1.11.2 Group of symmetries of a cube

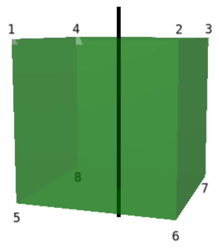
Demo on axis of symmetries of a cube
https://vimeo.com/90519329

```
In [130]: # cube(frame=False)

In [135]: P1 = vector([1,0,1])
          P2 = vector([1,1,1])
          P3 = vector([0,1,1])
          P4 = vector([0,0,1])
          P5 = vector([1,0,0])
          P6 = vector([1,1,0])
          P7 = vector([0,1,0])
          P8 = vector([0,0,0])
          S1 = polygon3d([P1,P2,P3,P4],color='green',opacity=0.5)
          S2 = polygon3d([P5,P6,P7,P8],color='green',opacity=0.5)
          S3 = polygon3d([P1,P2,P6,P5],color='green',opacity=0.5)
          S4 = polygon3d([P2,P3,P7,P6],color='green',opacity=0.5)
          S5 = polygon3d([P3,P4,P8,P7],color='green',opacity=0.5)
          S6 = polygon3d([P4,P1,P5,P8],color='green',opacity=0.5)
          t1 = text3d('1',(1,0,1.05))
          t2 = text3d('2',(1,1,1.05))
          t3 = text3d('3',(0,1,1.05))
          t4 = text3d('4',(0,0,1.05))
          t5 = text3d('5',(1,0,-0.1))
```

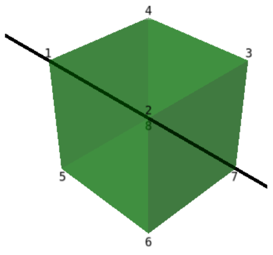
```
        t6 = text3d('6',(1,1,-0.1))
        t7 = text3d('7',(0,1,-0.1))
        t8 = text3d('8',(0,0,-0.1))
        cub = S1+S2+S3+S4+S5+S6+t1+t2+t3+t4+t5+t6+t7+t8
        show(cub,frame=False)
```



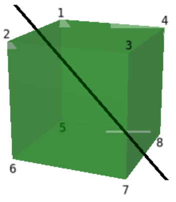
```
In [136]: var('t')
          ax1 = parametric_plot3d(t*(1/4*(P1+P2+P3+P4))+(1-t)*(1/4*(P5+P6+P7+P8)),\
                                  (t,-0.2,1.2),color='black',thickness=3)

          ax2 = parametric_plot3d(t*P1+(1-t)*P7,\
                                  (t,-0.2,1.2),color='black',thickness=3)
          ax3 = parametric_plot3d(t*1/2*(P1+P2)+(1-t)*1/2*(P7+P8),\
                                  (t,-0.2,1.2),color='black',thickness=3)

In [137]: show(cub+ax1,frame=False)
```



```
In [138]: show(cub+ax2,frame=False)
```



```
In [139]: show(cub+ax3,frame=False)
```



```
In [ ]: ## Cube has 13 axes of rotation of symmetry

In [140]: rot1 = "(1,2,3,4)(5,6,7,8)"
          rot2 = "(3,2,6,7)(4,1,5,8)"
          rot3 = "(1,2,6,5)(4,3,7,8)"
          Symm_cube = PermutationGroup([rot1,rot2,rot3 ])

In [141]: Symm_cube.order()

Out[141]: 24

In [142]: Symm_cube.list()

Out[142]: [(),
           (1,3)(2,4)(5,7)(6,8),
           (1,6)(2,5)(3,8)(4,7),
           (1,8)(2,7)(3,6)(4,5),
           (1,4,3,2)(5,8,7,6),
           (1,2,3,4)(5,6,7,8),
           (1,5)(2,8)(3,7)(4,6),
           (1,7)(2,6)(3,5)(4,8),
           (2,5,4)(3,6,8),
           (1,3,8)(2,7,5),
           (1,6,3)(4,5,7),
           (1,8,6)(2,4,7),
           (1,4)(2,8)(3,5)(6,7),
           (1,2,6,5)(3,7,8,4),
           (1,5,6,2)(3,4,8,7),
           (1,7)(2,3)(4,6)(5,8),
           (2,4,5)(3,8,6),
           (1,3,6)(4,7,5),
           (1,6,8)(2,7,4),
           (1,8,3)(2,5,7),
           (1,4,8,5)(2,3,7,6),
           (1,2)(3,5)(4,6)(7,8),
           (1,5,8,4)(2,6,7,3),
           (1,7)(2,8)(3,4)(5,6)]
```

```
In [145]: set([a.order() for a in Symm_cube])

Out[145]: {1, 2, 3, 4}

In [143]: Symm_cube.is_isomorphic(SymmetricGroup(4))

Out[143]: True

In [102]: Symm_cube.structure_description()

Out[102]: 'S4'
```

Assignments

1. Consider the group D_{12}. Find a normal subgroup H of G of order 6. Find the left and right cosets of H and show that they are equal. Also find the quotient group D_{12}/H.

2. Find normal subgroups H and K of order 6 and 12 in D_{24}. Verify the formula

$$|HK| = \frac{|H||K|}{|H \cap K|}.$$

3. Find all subgroups and all normal subgroups of D_{24}. Does D_{24} have a subgroup of order k for every divisor k of the order of D_{24}?

4. Draw a lattice of subgroups of cyclic group C_{30}. Label the vertices as order of the subgroup.

5. Find all Sylow subgroups of the dihedral group D(9).

6. Find all Sylow subgroups of the symmetric group S(6).

7. Find all Sylow subgroups of SL(2,5)
8. Find all Sylow p-subgroups of the alternating group A_5 and verify the third Sylow theorem.
9. Create user defined functions for fixed points, conjugacy classes, stabilizer etc for an action.
10. Construct the group of symmetries of a cube by labeling surfaces 1, 2,3,4,5,6, that is as a subgroup of $\mathbb{S}_6$.

```
In [ ]:
```

1.12 References

Some Suggested books on Sage

1. "Group Theory — An Expedition with SageMath"Ajit Kumar and Vikas Bist to be published by Narosa Publishing House, New Delhi.
2. "Calculus using Sage" by Sang-Gu Lee, Robert Beezer, Ajit Kumar and othes published by Kyungmoon Books
3. Online book on "Linear Algebra Sang-Gu Lee, Ajit Kumar and others.
4. Mathematical Computation with Sage by Paul Zimmermann
5. A First Course in Linear Algebra by Robert Beezer available online
6. Abstract Algebra: Theory and Applications by Tom Judson and Robert Beezer
7. An Introduction to SAGE Programming: With Applications to SAGE Interacts for Numerical Methods by Razvan A Mezei