In [1]:
```
import sympy
from sympy import  Point, Line, Plane, Point3D
```

Занятие 7

Алгебра

https://docs.sympy.org/latest/search.html?q=geometry

Взаимное расположение прямых и плоскостей в пространстве

Методы для изучения:

are_concurrent(*planes) определяет, пересекаются ли все плоскости по одной прямой, число плоскостей произвольное

are_concurrent(*lines) определяет, пересекаются ли все прямые в одной точке, число прямых произвольное

distance(o) расстояние между плоскостью и другим геометрическим объектом

equals(o) возвращает True, если self и o одинаковые математические объекты

intersection(o) пересечение с геометрическим объектом o

Пример 1.

Даны точки: $A(1,2,3)$, $B(-5,1,7)$, $C(3,-2,6)$. $D(-1,2,4)$. $F(5,1,7)$. $K(8,27,33)$. $M(16,24,32)$ $P(14,28,29)$.

Определить, пересекаются ли по общей прямой плоскости

а) ABC, ABK и ABF

б) ABC и AFK

Определить, пересекаются ли в одной точке прямые

а) AB и CD

б) AB и FK

In [2]:
```
A = Point(1, 2, 3)
B = Point(-5, 1, 7)
C = Point(3, -2, 6)
ABC = Plane(A, B, C)
D = Point(-1, 2, 4)
F = Point(5, 1, 7)
K = Point(8, 27, 33)
M = Point(16, 24, 32)
P = Point(14, 28, 29)
ABD = Plane(A, B, D)
ABF = Plane(A, B, F)
ABK = Plane(A, B, K)
AFK = Plane(A, F, K)
PKM = Plane(P, K, M)
AB = Line(A, B)
CD = Line(C, D)
FK = Line(F, K)
print('пересекаются ли по общей прямой ABC, ABK и ABF: ', Plane.are_concurrent(ABC, ABK, ABF))
print('пересекаются ли по общей прямой ABC и PKM: ', Plane.are_concurrent(ABC,PKM))
print('пересекаются ли в одной точке прямые  AB и CD: ', Line.are_concurrent(AB,CD))
print('пересекаются ли в одной точке прямые  AB и FK: ', Line.are_concurrent(AB,FK))
```

```
пересекаются ли по общей прямой ABC, ABK и ABF:  True
пересекаются ли по общей прямой ABC и PKM:  False
пересекаются ли в одной точке прямые  AB и CD:  True
пересекаются ли в одной точке прямые  AB и FK:  False
```

Пример 2

Найти расстояние до плоскости ABC:

а) от точки F

б) от прямой KM

в) от плоскости PKM

In [3]:
```
print('расстояние от  F до  ABC: ', ABC.distance(F))
KM = Line(K, M)
print('расстояние от  KM до  ABC: ', ABC.distance(KM))
print('расстояние от  PKM до  ABC: ', ABC.distance(PKM))
```

```
расстояние от  F до  ABC:  10/3
расстояние от  KM до  ABC:  39
расстояние от  PKM до  ABC:  39
```

Пример 3

Определить, совпадают ли плоскости

а) ABD и ABC

б) ABD и ABF

In [4]:
```
print(ABD.equals(ABC), ABD.equals(ABF))
```

```
True False
```

Пример 4

Найти пересечение:

а) плоскостей ABD и CKM

b) плоскости ABD и прямой FK

c) плоскости ABD и прямой KM

d) прямых AB и CD

In [5]:
```
CKM = Plane(C, K, M)
KM = Line(K, M)
print('ABD.intersection(CKM): ', *ABD.intersection(CKM))
print('ABD.intersection(FK): ', *ABD.intersection(FK))
print('ABD.intersection(KM): ', *ABD.intersection(KM))
print('AB.intersection(CD): ', *AB.intersection(CD))
```

```
ABD.intersection(CKM):  Line3D(Point3D(51, -20, 0), Point3D(987, -371, -117))
ABD.intersection(FK):  Point3D(505/107, -153/107, 489/107)
ABD.intersection(KM):
AB.intersection(CD):  Point3D(-5/7, 12/7, 29/7)
```

Плоскости ABD и CKM пересекаются по прямой, прямая FK пересекает плоскость ABD в точке, плоскость ABD не пересекается прямой KM, прямые AB и CD пересекаются в точке.

Проверим, что KM параллельна плоскости ABD:

In [6]:
```
print('ABD параллельно KM? ', ABD.is_parallel(KM))
```

```
ABD параллельно KM?  True
```

Пример 5

Работа со словарями

https://docs.python.org/3/c-api/dict.html?highlight=dictionary

Пример словаря:

создадим словари

а) всех точек

б) прямых, проходящих через эти точки

In [7]:
```
Points_list = [A, B, C, D]
Point_names = ['A','B','C','D']
points_numbers = range(len(Point_names))
Points = dict([(Point_names[i],Points_list[i]) for i in points_numbers])
Points
```

Out[7]:
```
{'A': Point3D(1, 2, 3),
 'B': Point3D(-5, 1, 7),
 'C': Point3D(3, -2, 6),
 'D': Point3D(-1, 2, 4)}
```

Вызовем точку A по ее имени:

In [8]:
```
Points['A']
```

Out[8]: $Point3D(1,2,3)$

Построим словарь прямых, проходящих через точки, используя присвоение по ключу.

(Присвоение по существующему ключу перезаписывает значение элемента словаря,

присвоение по несуществующему ключу добавляет новую запись в словарь)

In [9]:
```
Points.keys()
dict_keys(['A', 'B', 'C', 'D'])
```

Out[9]:
```
dict_keys(['A', 'B', 'C', 'D'])
```

In [10]:
```
lines = {P1 + P2: Line(Points[P1], Points[P2]) for P1 in Points.keys() for P2 in Points.keys() if P1 <
P2}
display(lines)
```

```
{'AB': Line3D(Point3D(1, 2, 3), Point3D(-5, 1, 7)),
 'AC': Line3D(Point3D(1, 2, 3), Point3D(3, -2, 6)),
 'AD': Line3D(Point3D(1, 2, 3), Point3D(-1, 2, 4)),
 'BC': Line3D(Point3D(-5, 1, 7), Point3D(3, -2, 6)),
 'BD': Line3D(Point3D(-5, 1, 7), Point3D(-1, 2, 4)),
 'CD': Line3D(Point3D(3, -2, 6), Point3D(-1, 2, 4))}
```

Пример 6

Даны точки пространстве $A(-2,7,-5)$. $B(6,2,-4)$. $C(14,-3,-2)$. $D(22,-8,-2)$. $K(30,-7,-4)$. $M(94,-35,-2)$.

Определить, какие тройки точек лежат на одной прямой, составить словарь с ключами - множествами из трех точек, значениями - прямыми Line.

Для проверки того, что три точки лежат на одной прямой используем Point3D.are_collinear.

Для иллюстрации возможности перебора по множеству будем выбирать первую точку из множества всех ключей словаря Points, вторую - из всех остальных точек, третью - из всех точек, кроме первой и второй. Если имена точек идут в лексикографическом порядке, и точки лежат на одной прямой, в словарь three_points включаем запись с ключом - tuple из имен трех точек, значением - прямой, проходящей через две из этих точек.

In [11]:
```
A = Point(-2, 7, -5)
B = Point(6, 2, -4)
C = Point(14, -3, -2)
D = Point(22, -8, -2)
F = Point(14, -3, -3)
K = Point(30, -7, -4)
M = Point(94, -35, -2)
Points = {A: 'A', B: 'B', C: 'C', D: 'D', F: 'F', K: 'K', M: 'M'}
three_points = {}
for P1 in Points.keys():
    for P2 in Points.keys() - {P1}:
        for P3 in Points.keys() - {P1} - {P2}:
            P_name1, P_name2, P_name3 = [Points[P] for P in (P1, P2, P3)]
            if P_name1 < P_name2 and P_name2 < P_name3:
                if Point3D.are_collinear(P1, P2, P3):
                    three_points[(P_name1, P_name2, P_name3)] = Line(P1, P2)
three_points
```

Out[11]:
```
{('A', 'D', 'F'): Line3D(Point3D(-2, 7, -5), Point3D(22, -8, -2)),
 ('A', 'B', 'D'): Line3D(Point3D(-2, 7, -5), Point3D(6, 2, -4)),
 ('A', 'B', 'F'): Line3D(Point3D(-2, 7, -5), Point3D(6, 2, -4)),
 ('A', 'K', 'M'): Line3D(Point3D(-2, 7, -5), Point3D(30, -7, -4)),
 ('B', 'D', 'F'): Line3D(Point3D(6, 2, -4), Point3D(22, -8, -2))}
```

Более оптимальный перебор вариантов с меньшим числом сравнений:

In [12]:
```
A = Point(-2, 7, -5)
B = Point(6, 2, -4)
C = Point(14, -3, -2)
D = Point(22, -8, -2)
F = Point(14, -3, -3)
K = Point(30, -7, -4)
M = Point(94, -35, -2)
Points_dict = {A: 'A', B: 'B', C: 'C', D: 'D', F: 'F', K: 'K', M: 'M'}
Points = tuple(Points_dict.keys())
num = len(Points)
three_points = {}
print('Точки на одной прямой')
for i in range(num):
    P1 = Points[i]
    for j in range(i + 1, num):
        P2 = Points[j]
        for k in range(j + 1, num):
            P3 = Points[k]
            P_name1, P_name2, P_name3 = [Points_dict[P] for P in (P1, P2, P3)]
            if P_name1 < P_name2 and P_name2 < P_name3:
                if Point3D.are_collinear(P1, P2, P3):
                    three_points[(P_name1, P_name2, P_name3)] = Line(P1, P2)
three_points
```

```
Точки на одной прямой
```

Out[12]:
```
{('A', 'B', 'D'): Line3D(Point3D(-2, 7, -5), Point3D(6, 2, -4)),
 ('A', 'B', 'F'): Line3D(Point3D(-2, 7, -5), Point3D(6, 2, -4)),
 ('A', 'D', 'F'): Line3D(Point3D(-2, 7, -5), Point3D(22, -8, -2)),
 ('A', 'K', 'M'): Line3D(Point3D(-2, 7, -5), Point3D(30, -7, -4)),
 ('B', 'D', 'F'): Line3D(Point3D(6, 2, -4), Point3D(22, -8, -2))}
```