```python
import sympy
from sympy import  Point, Line, Polygon, RegularPolygon
import matplotlib.pyplot as plt
```

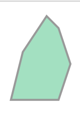
Занятие 9

Алгебра

Geometry: многоугольник на плоскости, принадлежность точки многоугольнику.

Пусть на плоскости задан многоугольник ABCDEF. Введем систему координат и создадим точки A,B,C,D,E и F. Из этих точек создадим многоугольник.

```python
A = Point(0, 0)
B = Point(4, 0)
C = Point(5, 3)
D = Point(4, 6)
E = Point(3, 7)
F = Point(1, 4)
Points = [A, B, C, D, E, F]
Poly = Polygon(*Points)
display(Poly)
```



Атрибуты многоугольника:

area площадь

angles углы

perimeter периметр

vertices вершины

centroid центр

sides стороны

bounds это tuple вида (xmin, ymin, xmax, ymax), определяющий прямоугольник, внутри которого помещается многоугольник

```python
display('площадь', Poly.area, 'углы', Poly.angles, 'периметр', Poly.perimeter, 'вершины', Poly.vertices
,
        'центр', Poly.centroid, ' стороны', Poly.sides, 'bounds', Poly.bounds)
```

'площадь'

$$\frac{45}{2}$$

'углы'

```
{Point2D(1, 4): acos(-14*sqrt(221)/221),
 Point2D(0, 0): acos(sqrt(17)/17),
 Point2D(4, 0): acos(-sqrt(10)/10),
 Point2D(5, 3): acos(-4/5),
 Point2D(4, 6): acos(-2*sqrt(5)/5),
 Point2D(3, 7): acos(sqrt(26)/26)}
```

'периметр'

$$\sqrt{2}+\sqrt{13}+4+\sqrt{17}+2\sqrt{10}$$

'вершины'

```
[Point2D(0, 0),
 Point2D(4, 0),
 Point2D(5, 3),
 Point2D(4, 6),
 Point2D(3, 7),
 Point2D(1, 4)]
```

'центр'

$$Point2D\left(\frac{8}{3}, \frac{383}{135}\right)$$

' стороны'

```
[Segment2D(Point2D(0, 0), Point2D(4, 0)),
 Segment2D(Point2D(4, 0), Point2D(5, 3)),
 Segment2D(Point2D(5, 3), Point2D(4, 6)),
 Segment2D(Point2D(4, 6), Point2D(3, 7)),
 Segment2D(Point2D(3, 7), Point2D(1, 4)),
 Segment2D(Point2D(1, 4), Point2D(0, 0))]
```

'bounds'

```
(0, 0, 5, 7)
```

Для более привычного отображения многоугольника используем словарь с ключами - точками, значениями - именами точек:

```python
point_dict = {A: 'A', B: 'B', C: 'C', D: 'D', E: 'E', F: 'F'}
print('вершины: ', *[point_dict[vert] for vert in Poly.vertices])
```

```
вершины:  A B C D E F
```

Выведем стороны многоугольника в виде имен, к концам сегмента получаем доступ через свойства p1 и p2 класса Segment:

```python
print(*[point_dict[side.p1] + point_dict[side.p2] for side in Poly.sides])
```

```
AB BC CD DE EF FA
```

Выведем углы, используя словарь, будем выводить формулы углов, приближенные значения в радианах и в градусах:

```python
angles = Poly.angles
for point in angles.keys():
    angle = angles[point]
    print('Угол ', point_dict[point], ' = ', angle, ' = ', angle.evalf(3), ' рад = ', round(angle*180/s
ympy.pi), 'град')
```

```
Угол  F  =  acos(-14*sqrt(221)/221)  =  2.80  рад =  160 град
Угол  A  =  acos(sqrt(17)/17)  =  1.33  рад =  76 град
Угол  B  =  acos(-sqrt(10)/10)  =  1.89  рад =  108 град
Угол  C  =  acos(-4/5)  =  2.50  рад =  143 град
Угол  D  =  acos(-2*sqrt(5)/5)  =  2.68  рад =  153 град
Угол  E  =  acos(sqrt(26)/26)  =  1.37  рад =  79 град
```

Методы многоугольников:

cut_section(line) возвращает tuple из двух частей многоугольника, лежащих выше и ниже прямой line

distance(o) возвращает расстояние между self и o (если o - точка, то self не обязан быть выпуклым, если o - многоугольник, то self и o обязательно выпуклые)

encloses_point(p) возвращает True, если p - внутренняя точка многоугольника self, граничные точки дают False.

intersection(o) возвращает список из общих частей self и o

is_convex() возвращает True, если многоугольник выпуклый

```python
AD = Line(A, D)
print('cut_section(AD)\n', Poly.cut_section(AD)[0], '\n', Poly.cut_section(AD)[1],
      '\ndistance(Point(10,10))', Poly.distance(Point(10,10)),
      '\ndistance(A)', Poly.distance(A), '\nencloses_point((A + D)/2)', Poly.encloses_point((A + D)/2),
      '\nencloses_point(D)', Poly.encloses_point(D), '\nintersection(AD)', Poly.intersection(AD),
      '\nis_convex', Poly.is_convex())
```

```
cut_section(AD)
 Polygon(Point2D(4, 6), Point2D(3, 7), Point2D(1, 4), Point2D(0, 0))
 Polygon(Point2D(0, 0), Point2D(4, 0), Point2D(5, 3), Point2D(4, 6))
distance(Point(10,10)) 2*sqrt(13)
distance(A) 0
encloses_point((A + D)/2) True
encloses_point(D) False
intersection(AD) [Point2D(0, 0), Point2D(4, 6)]
is_convex True
```

Правильные многоугольники RegularPolygon

Правильный многоугольник --- равносторонний с равными внутренними углами

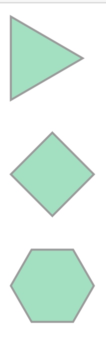
Параметры правильного многоугольника:

center центр

radius радиус описанной окружности, равен расстоянию от центра до любой из вершин

n число сторон (углов)

```python
RPoly3 = RegularPolygon(A, 5, 3)
display(RPoly3)
RPoly4 = RegularPolygon(A, 5, 4)
display(RPoly4)
RPoly6 = RegularPolygon(A, 5, 6)
display(RPoly6)
```



Атрибуты правильного многоугольника:

vertices вершины

center центр

radius радиус

rotation поворот

apothem радиус вписанной окружности

interior_angle внутренний угол

exterior_angle внешний угол

circumcircle описанная окружность

incircle вписанная окружность

angles углы

Для вывода имен вершин правильного многоугольника создадим словарь последним элементов включим в этот словарь центр мнгоугольника.

```python
RPoly = RegularPolygon(A, 5, 8)
point_RPoly = {vert: 'A' + str(i + 1) for i, vert in enumerate(RPoly.vertices + [RPoly.center])}
angles = RPoly.angles
print('вершины', *[point_RPoly[vert] for vert in RPoly.vertices],
      '\nцентр', point_RPoly[RPoly.center],
      '\nрадиус', RPoly.radius,
      '\nповорот', RPoly.rotation,
      '\nрадиус вписанной окружности', RPoly.apothem,
      '\nвнутренний угол', RPoly.interior_angle,
      '\nвнешний угол', RPoly.exterior_angle,
      '\nописанная окружность',  RPoly.circumcircle,
      '\nвписанная окружность', RPoly.incircle,
      '\nуглы', *[point_RPoly[key] + ' = ' + str(angles[key]) + ' ' for key in angles.keys()])
```

```
вершины A1 A2 A3 A4 A5 A6 A7 A8
центр A9
радиус 5
поворот 0
радиус вписанной окружности 5*sqrt(sqrt(2)/4 + 1/2)
внутренний угол 3*pi/4
внешний угол pi/4
описанная окружность Circle(Point2D(0, 0), 5)
вписанная окружность Circle(Point2D(0, 0), 5*sqrt(sqrt(2)/4 + 1/2))
углы A1 = 3*pi/4  A2 = 3*pi/4  A3 = 3*pi/4  A4 = 3*pi/4  A5 = 3*pi/4  A6 = 3*pi/4  A7 = 3*pi/4  A8 =
3*pi/4
```

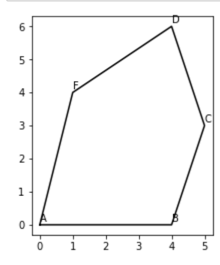
Изображение многоугольников

Опишем в виде функции построение многоугольника на графике:

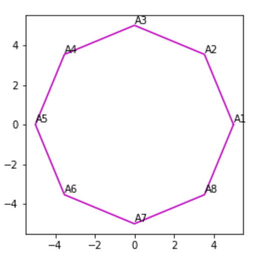
```python
def draw_polygon(polygon, names=None, color='g'):
    ax = plt.gca()
    ax.set_aspect('equal')
    vertices = polygon.vertices
    points_x = [item.x for item in vertices] + [vertices[0].x]
    points_y = [item.y for item in vertices] + [vertices[0].y]
    ax.plot(points_x, points_y, color)
    if names == None:
        for i, x_coord in enumerate(points_x[:-1]):
            ax.annotate('A' + str(i + 1),
                xy=(x_coord, points_y[i] + 0.1), xycoords='data') # Координаты подписываемой точки, туда ве
дет стрелка
    else:
        for i, vert in enumerate(vertices):
            ax.annotate(names[vert],
                xy=(points_x[i], points_y[i] + 0.1), xycoords='data') # Координаты подписываемой точки, туд
а ведет стрелка
```

Изобразим многоугольник с вершинами A, B, C, D, F

```python
draw_polygon(Polygon(A, B, C, D, F), names=point_dict, color='k')
```



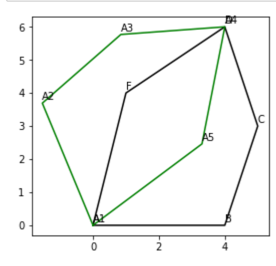

```python
draw_polygon(RPoly, color='m')
```



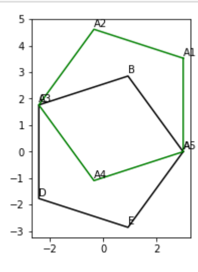
Преобразования могоугольников

reflect(line) - отражение относительно прямой

```python
Poly5 = Polygon(A, B, C, D, F)
draw_polygon(Poly5, names=point_dict, color='k')
draw_polygon(Poly5.reflect(AD))
```

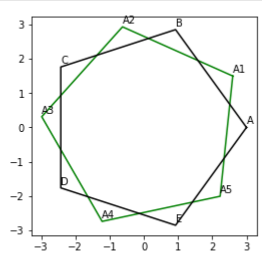



```python
RPoly5 = RegularPolygon(A, 3, 5)
letter = ['A', 'B', 'C', 'D', 'E', 'O']
vertices = RPoly5.vertices
point_RPoly = {vert: letter[i] for i, vert in enumerate(vertices + [RPoly5.center])}
draw_polygon(RPoly5, names=point_RPoly, color='k')
draw_polygon(RPoly5.reflect(Line(vertices[0], vertices[2])))
```



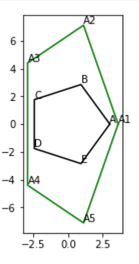
rotate(angle, pt=None) поворот на угол angle, сам многоугольник не изменяется!

```python
draw_polygon(RPoly5.rotate(sympy.pi/6))
draw_polygon(RPoly5, names=point_RPoly, color='k')
```



scale(x=1, y=1, pt=None) растяжение - сжатие

```python
draw_polygon(RPoly5, names=point_RPoly, color='k')
draw_polygon(RPoly5.scale(1.2, 2.5))
```



spin(angle) поворачивает сам многоугольник, а не копию

```python
draw_polygon(RPoly5, names=point_RPoly, color='k')
RPoly5.spin(sympy.pi/3)
draw_polygon(RPoly5)
```