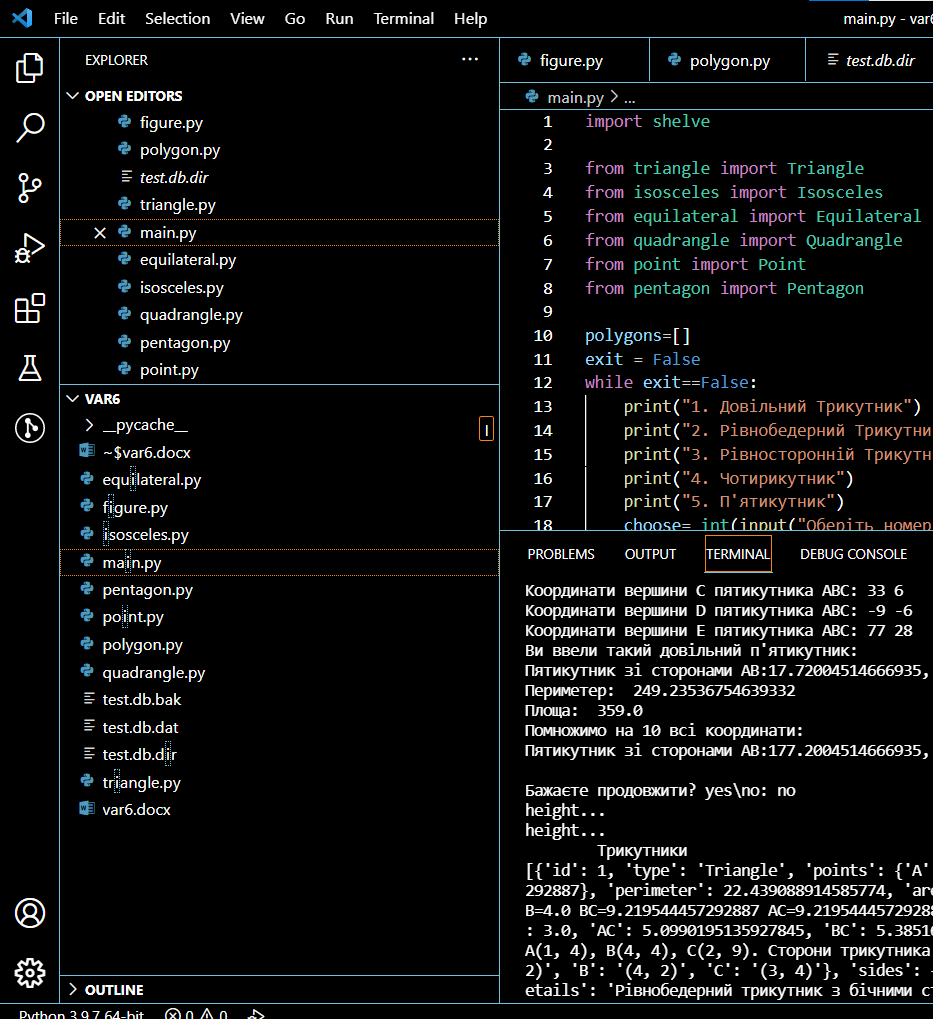


Структура проекту:



Модуль figure:

class Figure:

    def \_\_init\_\_(self, angles):

        self.angles = angles

    def \_\_str\_\_(self):

        return "Фігура"

    def cnt\_angles(self):

        return self.angles

Модуль polygon:

from point import Point

from figure import Figure

from math import sqrt

class Polygon(Figure):

    def \_\_init\_\_(self, angles):

        Figure.\_\_init\_\_(self, angles)

    def whoamI(self):

        if self.angles == 3:

            return "Трикутник"

        elif self.angles == 4:

            return "Чотирикутник"

        elif self.angles == 5:

            return "Пятикутник"

        else: return "Багатокутник"

    def area(self):

        pass

    def perimeter(self):

        pass

    def sideLen(self, first: Point, second: Point):

        return sqrt( (second.x-first.x)\*\*2 + (second.y-first.y)\*\*2 )

    def \_\_str\_\_(self):

        return "Багатокутник"

Модуль point:

class Point:

    def \_\_init\_\_(self, x : float, y : float):

        self.x=x

        self.y=y

    def \_\_str\_\_(self):

        return "({0}, {1})".format(self.x,self.y)

    def \_\_mul\_\_(self, other):

        x = self.x\*other

        y = self.y\*other

        return Point(x,y)

    def \_\_eq\_\_(self, other):

        return self.x==other.x and self.y==other.y

Модуль triangle:

from polygon import Polygon

from point import Point

from math import sqrt

class Triangle(Polygon):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point):

        Polygon.\_\_init\_\_(self, 3)

        self.A = a

        self.B = b

        self.C = c

        self.len\_AB = self.sideLen(self.A, self.B)

        self.len\_AC = self.sideLen(self.A, self.C)

        self.len\_BC = self.sideLen(self.B, self.C)

    def area(self):

        p = self.perimeter()/2

        return round(sqrt(p\*(p-self.len\_AB)\*(p-self.len\_AC)\*(p-self.len\_BC)) ,2)

    def perimeter(self):

        return self.len\_AB+self.len\_AC+self.len\_BC

    def \_\_mul\_\_(self, other):

        a = self.A \* other

        b = self.B \* other

        c = self.C \* other

        return Triangle(a, b, c)

    def \_\_eq\_\_(self, other):

        return self.A==other.A and self.B==other.B and self.C==other.C

    def \_\_str\_\_(self):

        return "Трикутник АВС. Вершини трикутника: A{0}, B{1}, C{2}. Сторони трикутника: AB={3} BC={4} AC={5}".format(self.A, self.B, self.C, self.len\_AB, self.len\_BC, self.len\_AC)

Модуль isoscales:

from triangle import Triangle

from point import Point

from math import sqrt

class Isosceles(Triangle):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point):

        Triangle.\_\_init\_\_(self, a, b, c)

    def heigth(self):

        return self.area()/self.len\_AC

    def \_\_str\_\_(self):

        if self.len\_BC==self.len\_AB:

            return ("Рівнобедерний трикутник з бічними сторонами: {0} та {1} та основою: {2}".format(self.len\_AB, self.len\_BC, self.len\_AC))

        elif self.len\_BC==self.len\_AC:

            return ("Рівнобедерний трикутник з бічними сторонами: {0} та {1} та основою: {2}".format(self.len\_AC, self.len\_BC, self.len\_AB))

        else:

            return ("Рівнобедерний трикутник з бічними сторонами: {0} та {1} та основою: {2}".format(self.len\_AB, self.len\_AC, self.len\_BC))

Модуль equilateral:

from triangle import Triangle

from point import Point

import math

class Equilateral(Triangle):

    def \_\_init\_\_(self,a: Point, b: Point, c: Point):

        Triangle.\_\_init\_\_(self, a, b, c)

    def area(self):

        return (math.pow(self.len\_AB,2)\*(math.sqrt(3)))/4

    def height(self):

        return (math.sqrt(3)\*self.len\_AB)/2

    def \_\_str\_\_(self):

        return ("Рівносторонній трикутник з трьома рівними сторонами сторонами: {0}".format(self.len\_AB))

Модуль quadrangle:

from polygon import Polygon

from point import Point

from math import sqrt

class Quadrangle(Polygon):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point, d: Point):

        Polygon.\_\_init\_\_(self, 4)

        self.A = a

        self.B = b

        self.C = c

        self.D = d

        self.len\_AB = self.sideLen(self.A, self.B)

        self.len\_BC = self.sideLen(self.B, self.C)

        self.len\_CD = self.sideLen(self.C, self.D)

        self.len\_AD = self.sideLen(self.A, self.D)

    def area(self):

        p = self.perimeter()/2

        area = sqrt((p-self.len\_AB)\*(p-self.len\_CD)\*(p-self.len\_AD)\*(p-self.len\_BC))

        return area

    def perimeter(self):

        return self.len\_AB + self.len\_AD + self.len\_BC + self.len\_CD

    def \_\_mul\_\_(self, other):

        a = self.A \* other

        b = self.B \* other

        c = self.C \* other

        d = self.D \* other

        return Quadrangle(a, b, c, d)

    def \_\_eq\_\_(self, other):

        return self.A==other.A and self.B==other.B and self.C==other.C and self.D==other.D

    def \_\_str\_\_(self):

        return "Чотирикутник з координатами А{0} B{1} C{2} D{3} та зі сторонами AB:{4}, BC:{5}, AD:{6}, CD:{7}".format(self.A, self.B, self.C, self.D, self.len\_AB, self.len\_BC, self.len\_AD, self.len\_CD)

Модуль pentagon:

from polygon import Polygon

from point import Point

from triangle import Triangle

class Pentagon(Polygon):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point, d: Point, e: Point):

        Polygon.\_\_init\_\_(self, 5)

        self.A = a

        self.B = b

        self.C = c

        self.D = d

        self.E = e

        self.len\_AB = self.sideLen(self.A, self.B)

        self.len\_BC = self.sideLen(self.B, self.C)

        self.len\_CD = self.sideLen(self.C, self.D)

        self.len\_DE = self.sideLen(self.E, self.D)

        self.len\_AE = self.sideLen(self.A, self.E)

    def area(self):

        t1 = Triangle(self.A, self.B, self.C)

        t2 = Triangle(self.A, self.C, self.D)

        t3 = Triangle(self.A, self.D, self.E)

        t4 = Triangle(self.A, self.B, self.E)

        area = t1.area() + t2.area() + t3.area() + t4.area()

        return area

    def perimeter(self):

        return self.len\_AB + self.len\_AE + self.len\_BC + self.len\_CD + self.len\_DE

    def \_\_mul\_\_(self, other):

        a = self.A \* other

        b = self.B \* other

        c = self.C \* other

        d = self.D \* other

        e = self.E \* other

        return Pentagon(a, b, c, d, e)

    def \_\_eq\_\_(self, other):

        return self.A==other.A and self.B==other.B and self.C==other.C and self.D==other.D and self.E == other.E

    def \_\_str\_\_(self):

        return "Пятикутник зі сторонами AB:{0}, BC:{1}, CD:{2}, DE:{3}, AE:{4}".format(self.len\_AB, self.len\_BC, self.len\_CD, self.len\_DE, self.len\_AE)

Головний модуль main:

import shelve

from triangle import Triangle

from isosceles import Isosceles

from equilateral import Equilateral

from quadrangle import Quadrangle

from point import Point

from pentagon import Pentagon

polygons=[]

exit = False

while exit==False:

    print("1. Довільний Трикутник")

    print("2. Рівнобедерний Трикутник")

    print("3. Рівносторонній Трикутник")

    print("4. Чотирикутник")

    print("5. П'ятикутник")

    choose= int(input("Оберіть номер фігури від 1 до 5 : "))

    if choose==1:

        x1, y1 = [int(x) for x in input("Координати вершини А трикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В трикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С трикутника АВС: ").split()]

        triangle = Triangle(Point(x1,y1), Point(x2, y2), Point(x3, y3))

        polygons.append(triangle)

        print("Ви ввели такий довільний трикутник: ")

        print(triangle)

        print("Периметер: ",triangle.perimeter())

        print("Площа: ", triangle.area())

        print("Помножимо на 3 всі координати:")

        print(triangle \* 3)

        print()

    elif choose==2:

        x1, y1 = [int(x) for x in input("Координати вершини А трикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В трикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С трикутника АВС: ").split()]

        isosceles = Isosceles(Point(x1,y1), Point(x2, y2), Point(x3, y3))

        polygons.append(isosceles)

        print("Ви ввели такий рівнобедерний трикутник: ")

        print(isosceles)

        print("Периметер: ",isosceles.perimeter())

        print("Площа: ", isosceles.area())

        print("Помножимо на 8 всі координати:")

        print(isosceles \* 8)

        print()

    elif choose==3:

        x1, y1 = [int(x) for x in input("Координати вершини А трикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В трикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С трикутника АВС: ").split()]

        equilateral = Equilateral(Point(x1,y1), Point(x2, y2), Point(x3, y3))

        polygons.append(equilateral)

        print("Ви ввели такий рівносторонній трикутник: ")

        print(equilateral)

        print("Периметер: ",equilateral.perimeter())

        print("Площа: ", equilateral.area())

        print("Помножимо на 5 всі координати:")

        print(equilateral \* 5)

        print()

    elif choose==4:

        x1, y1 = [int(x) for x in input("Координати вершини А чотирикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В чотирикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С чотирикутника АВС: ").split()]

        x4, y4 = [int(x) for x in input("Координати вершини D чотирикутника АВС: ").split()]

        quadrangle = Quadrangle(Point(x1,y1), Point(x2, y2), Point(x3, y3),Point(x4, y4))

        polygons.append(quadrangle)

        print("Ви ввели такий довільний чотирикутник: ")

        print(quadrangle)

        print("Периметер: ",quadrangle.perimeter())

        print("Площа: ", quadrangle.area())

        print("Помножимо на 2 всі координати:")

        print(quadrangle \* 2)

        print()

    elif choose==5:

        x1, y1 = [int(x) for x in input("Координати вершини А пятикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В пятикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С пятикутника АВС: ").split()]

        x4, y4 = [int(x) for x in input("Координати вершини D пятикутника АВС: ").split()]

        x5, y5 = [int(x) for x in input("Координати вершини E пятикутника АВС: ").split()]

        pentagon = Pentagon(Point(x1,y1), Point(x2, y2), Point(x3, y3),Point(x4, y4), Point(x5, y5))

        polygons.append(pentagon)

        print("Ви ввели такий довільний п'ятикутник: ")

        print(pentagon)

        print("Периметер: ",pentagon.perimeter())

        print("Площа: ", pentagon.area())

        print("Помножимо на 10 всі координати:")

        print(pentagon \* 10)

        print()

    answer = input("Бажаєте продовжити? yes\\no: ")

    if answer.lower()=='n' or answer.lower()=='no':

        exit=True

triangle1 = Triangle(Point(1,2), Point(5, 2), Point(3, 11))

triangle2 = Triangle(Point(1,4), Point(4, 4), Point(2, 9))

isoscele1 = Isosceles(Point(2,2), Point(4, 2), Point(3, 4))

quadrangle1 = Quadrangle(Point(-1,-10), Point(2, 4), Point(3,7),Point(5, 6))

pentagon1 = Pentagon(Point(-8,-5), Point(5, 4), Point(2,37),Point(4, 8), Point(12,34))

polygons.append(triangle1)

polygons.append(triangle2)

polygons.append(isoscele1)

polygons.append(quadrangle1)

polygons.append(pentagon1)

with shelve.open('test.db') as db:

    triangles=[]

    pentagons=[]

    quadrangles=[]

    id\_triangle = 1

    id\_quadrangle = 1

    id\_pentagon = 1

    for ob in polygons:

        if type(ob)==Triangle or type(ob)==Isosceles or type(ob)==Equilateral:

            js = {

            'id': id\_triangle,

            'angles':ob.angles,

            'type' : str(type(ob).\_\_name\_\_),

            'points': {'A': str(ob.A), 'B': str(ob.B), 'C': str(ob.C)},

            'sides': {'AB':ob.len\_AB, 'AC':ob.len\_AC, 'BC':ob.len\_BC},

            'perimeter':ob.perimeter(),

            'area':ob.area(),

            'details': str(ob)

            }

            if (type(ob)==Isosceles):

                js['height']=ob.heigth()

            triangles.append(js)

            id\_triangle+=1

        elif type(ob)==Quadrangle:

            quadrangles.append( {

            'id': id\_quadrangle,

            'angles':ob.angles,

            'points': {'A': str(ob.A), 'B': str(ob.B), 'C': str(ob.C), 'D':str(ob.D)},

            'sides': {'AB':ob.len\_AB, 'BC':ob.len\_BC, 'CD':ob.len\_CD, 'AD': ob.len\_AD},

            'perimeter':ob.perimeter(),

            'area':ob.area(),

            'details': str(ob) })

            id\_quadrangle+=1

        elif type(ob)==Pentagon:

            pentagons.append( {

            'id': id\_pentagon,

            'angles':ob.angles,

            'points': {'A': str(ob.A), 'B': str(ob.B), 'C': str(ob.C), 'D':str(ob.D), 'E':str(ob.E)},

            'sides': {'AB':ob.len\_AB, 'BC':ob.len\_BC, 'CD':ob.len\_CD, 'DE': ob.len\_DE, 'AE': ob.len\_AE},

            'perimeter':ob.perimeter(),

            'area':ob.area(),

            'details': str(ob) })

            id\_pentagon+=1

    db['Triangles'] = triangles

    db['Quadrangles'] = quadrangles

    db['Pentagons'] = pentagons

with shelve.open('test.db') as db:

    print("\tТрикутники")

    print(db['Triangle'])

    print("\tЧотирикутники")

    print(db['Quadrangle'])

    print("\tП'ятикутники")

    print(db['Pentagons'])

Створено класи

1. Figure - базовий клас, у якого є атрибут angles, що відповідає за кількість кутів фігури.

Конструктор класу Figure:

class Figure:

    def \_\_init\_\_(self, angles):

        self.angles = angles

1. Polygon - багатокутник, клас наслідується від батьківського Figure.

Конструктор класу Polygon :

class Polygon(Figure):

    def \_\_init\_\_(self, angles):

        Figure.\_\_init\_\_(self, angles)

1. Point - точка, у якої є два атрибути x та y, що відповідають за координати.

Конструктор класу Point :

class Point:

    def \_\_init\_\_(self, x : float, y : float):

        self.x=x

        self.y=y

1. Triangle - трикутник, клас наслідується від батьківського Polygon. У трикутника є атрибути A, B, C - точки трикутника та атрибути len\_AB, len\_BC, len\_AC - сторони трикутника.

Конструктор класу Triangle :

class Triangle(Polygon):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point):

        Polygon.\_\_init\_\_(self, 3)

        self.A = a

        self.B = b

        self.C = c

        self.len\_AB = self.sideLen(self.A, self.B)

        self.len\_AC = self.sideLen(self.A, self.C)

        self.len\_BC = self.sideLen(self.B, self.C)

1. Isoscales - рівнобедерний трикутник, клас наслідується від батьківського Triangle.

Конструктор класу Isoscales :

class Isosceles(Triangle):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point):

        Triangle.\_\_init\_\_(self, a, b, c)

1. Equilateral - рівносторонній трикутник, клас наслідується від батьківського Triangle

Конструктор класу Equilateral :

class Equilateral(Triangle):

    def \_\_init\_\_(self,a: Point, b: Point, c: Point):

        Triangle.\_\_init\_\_(self, a, b, c)

1. Quadrangle - чотирикутник, клас наслідується від батьківського Polygon. У чотирикутника є атрибути A, B, C, D - точки чотирикутника, та атрибути len\_AB, len\_BC, len\_BD, len\_AD - сторони чотирикутника.

Конструктор класу Quadrangle :

class Quadrangle(Polygon):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point, d: Point):

        Polygon.\_\_init\_\_(self, 4)

        self.A = a

        self.B = b

        self.C = c

        self.D = d

        self.len\_AB = self.sideLen(self.A, self.B)

        self.len\_BC = self.sideLen(self.B, self.C)

        self.len\_CD = self.sideLen(self.C, self.D)

        self.len\_AD = self.sideLen(self.A, self.D)

1. Pentagon - пятикутник, клас наслідується від батьківського Polygon. У пятикутника є атрибути A, B, C, D, E - точки пятикутника, та атрибути len\_AB, len\_BC, len\_BD, len\_DE, len\_AE - сторони пятикутника.

Конструктор класу Pentagon :

class Pentagon(Polygon):

    def \_\_init\_\_(self, a: Point, b: Point, c: Point, d: Point, e: Point):

        Polygon.\_\_init\_\_(self, 5)

        self.A = a

        self.B = b

        self.C = c

        self.D = d

        self.E = e

        self.len\_AB = self.sideLen(self.A, self.B)

        self.len\_BC = self.sideLen(self.B, self.C)

        self.len\_CD = self.sideLen(self.C, self.D)

        self.len\_DE = self.sideLen(self.E, self.D)

        self.len\_AE = self.sideLen(self.A, self.E)

В усіх підкласах визначено адаптивні конструктори. Методи, що визначають поведінку класів. У класі Figure є метод cnt\_angles, що повертає кількість кутів фігури:

    def cnt\_angles(self):

        return self.angles

У класі Polygon є метод whoamI, який повертає рядок з назвою багатокутника за кількістю його сторін:

    def whoamI(self):

        if self.angles == 3:

            return "Трикутник"

        elif self.angles == 4:

            return "Чотирикутник"

        elif self.angles == 5:

            return "Пятикутник"

        else: return "Багатокутник"

Також у класі Polygon є метод sideLen, який за координатами визначає довжину сторони:

    def sideLen(self, first: Point, second: Point):

        return sqrt( (second.x-first.x)\*\*2 + (second.y-first.y)\*\*2 )

Також у класі Polygon є невизначені методи area та perimeter

    def area(self):

        pass

    def perimeter(self):

        pass

У класі Triangle визначено методи для обрахунку площі та периметра трикутника:

    def area(self):

        p = self.perimeter()/2

        return round(sqrt(p\*(p-self.len\_AB)\*(p-self.len\_AC)\*(p-self.len\_BC)) ,2)

    def perimeter(self):

        return self.len\_AB+self.len\_AC+self.len\_BC

У класі Equliteral перевизначено метод для обрахунку площі:

    def area(self):

        return (math.pow(self.len\_AB,2)\*(math.sqrt(3)))/4

У класі Quadrangle перевизначено методи для обрахунку площі та периметру чотирикутника:

    def area(self):

        p = self.perimeter()/2

        area = sqrt((p-self.len\_AB)\*(p-self.len\_CD)\*(p-self.len\_AD)\*(p-self.len\_BC))

        return area

    def perimeter(self):

        return self.len\_AB + self.len\_AD + self.len\_BC + self.len\_CD

У класі Pentagon перевизначено методи для обрахунку площі та периметру пятикутника. Оскільки пятикутник не обовязково має бути рівностороннім, то використовується формула площі через орієнтовані трикутники:

    def area(self):

        t1 = Triangle(self.A, self.B, self.C)

        t2 = Triangle(self.A, self.C, self.D)

        t3 = Triangle(self.A, self.D, self.E)

        t4 = Triangle(self.A, self.B, self.E)

        area = t1.area() + t2.area() + t3.area() + t4.area()

        return area

    def perimeter(self):

        return self.len\_AB + self.len\_AE + self.len\_BC + self.len\_CD + self.len\_DE

У класі Point перегружено оператори \_\_mul\_\_, \_\_eq\_\_, \_\_str\_\_:

    def \_\_str\_\_(self):

        return "({0}, {1})".format(self.x,self.y)

    def \_\_mul\_\_(self, other):

        x = self.x\*other

        y = self.y\*other

        return Point(x,y)

    def \_\_eq\_\_(self, other):

        return self.x==other.x and self.y==other.y

У класі Figure перегружено оператор\_\_str\_\_:

    def \_\_str\_\_(self):

        return "Фігура"

У класі Polygon перегружено оператор \_\_str\_\_:

    def \_\_str\_\_(self):

        return "Багатокутник"

У класі Triangle перегружено оператори \_\_mul\_\_, \_\_eq\_\_, \_\_str\_\_:

    def \_\_mul\_\_(self, other):

        a = self.A \* other

        b = self.B \* other

        c = self.C \* other

        return Triangle(a, b, c)

    def \_\_eq\_\_(self, other):

        return self.A==other.A and self.B==other.B and self.C==other.C

    def \_\_str\_\_(self):

        return "Трикутник АВС. Вершини трикутника: A{0}, B{1}, C{2}. Сторони трикутника: AB={3} BC={4} AC={5}".format(self.A, self.B, self.C, self.len\_AB, self.len\_BC, self.len\_AC)

У класі Equliteral перегружено оператор \_\_str\_\_:

    def \_\_str\_\_(self):

        return ("Рівносторонній трикутник з трьома рівними сторонами сторонами: {0}".format(self.len\_AB))

У класі Isoscales перегружено оператор \_\_str\_\_:

    def \_\_str\_\_(self):

        if self.len\_BC==self.len\_AB:

            return ("Рівнобедерний трикутник з бічними сторонами: {0} та {1} та основою: {2}".format(self.len\_AB, self.len\_BC, self.len\_AC))

        elif self.len\_BC==self.len\_AC:

            return ("Рівнобедерний трикутник з бічними сторонами: {0} та {1} та основою: {2}".format(self.len\_AC, self.len\_BC, self.len\_AB))

        else:

            return ("Рівнобедерний трикутник з бічними сторонами: {0} та {1} та основою: {2}".format(self.len\_AB, self.len\_AC, self.len\_BC))

У класі Quadrangle перегружено оператори \_\_mul\_\_, \_\_eq\_\_, \_\_str\_\_:

    def \_\_mul\_\_(self, other):

        a = self.A \* other

        b = self.B \* other

        c = self.C \* other

        d = self.D \* other

        return Quadrangle(a, b, c, d)

    def \_\_eq\_\_(self, other):

        return self.A==other.A and self.B==other.B and self.C==other.C and self.D==other.D

    def \_\_str\_\_(self):

        return "Чотирикутник з координатами А{0} B{1} C{2} D{3} та зі сторонами AB:{4}, BC:{5}, AD:{6}, CD:{7}".format(self.A, self.B, self.C, self.D, self.len\_AB, self.len\_BC, self.len\_AD, self.len\_CD)

У класі Pentagon перегружено оператори \_\_mul\_\_, \_\_eq\_\_, \_\_str\_\_:

    def \_\_mul\_\_(self, other):

        a = self.A \* other

        b = self.B \* other

        c = self.C \* other

        d = self.D \* other

        e = self.E \* other

        return Pentagon(a, b, c, d, e)

    def \_\_eq\_\_(self, other):

        return self.A==other.A and self.B==other.B and self.C==other.C and self.D==other.D and self.E == other.E

    def \_\_str\_\_(self):

        return "Пятикутник зі сторонами AB:{0}, BC:{1}, CD:{2}, DE:{3}, AE:{4}".format(self.len\_AB, self.len\_BC, self.len\_CD, self.len\_DE, self.len\_AE)

Крок 6. Збереження об'єктів в базі даних (модулі shelve і dbm). Дослідження сховища в інтерактивному сеансі

Реалізовано введення даних з клавіатури та виведення у консоль:

polygons=[]

exit = False

while exit==False:

    print("1. Довільний Трикутник")

    print("2. Рівнобедерний Трикутник")

    print("3. Рівносторонній Трикутник")

    print("4. Чотирикутник")

    print("5. П'ятикутник")

    choose= int(input("Оберіть номер фігури від 1 до 5 : "))

    if choose==1:

        x1, y1 = [int(x) for x in input("Координати вершини А трикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В трикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С трикутника АВС: ").split()]

        triangle = Triangle(Point(x1,y1), Point(x2, y2), Point(x3, y3))

        polygons.append(triangle)

        print("Ви ввели такий довільний трикутник: ")

        print(triangle)

        print("Периметер: ",triangle.perimeter())

        print("Площа: ", triangle.area())

        print("Помножимо на 3 всі координати:")

        print(triangle \* 3)

        print()

    elif choose==2:

        x1, y1 = [int(x) for x in input("Координати вершини А трикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В трикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С трикутника АВС: ").split()]

        isosceles = Isosceles(Point(x1,y1), Point(x2, y2), Point(x3, y3))

        polygons.append(isosceles)

        print("Ви ввели такий рівнобедерний трикутник: ")

        print(isosceles)

        print("Периметер: ",isosceles.perimeter())

        print("Площа: ", isosceles.area())

        print("Помножимо на 8 всі координати:")

        print(isosceles \* 8)

        print()

    elif choose==3:

        x1, y1 = [int(x) for x in input("Координати вершини А трикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В трикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С трикутника АВС: ").split()]

        equilateral = Equilateral(Point(x1,y1), Point(x2, y2), Point(x3, y3))

        polygons.append(equilateral)

        print("Ви ввели такий рівносторонній трикутник: ")

        print(equilateral)

        print("Периметер: ",equilateral.perimeter())

        print("Площа: ", equilateral.area())

        print("Помножимо на 5 всі координати:")

        print(equilateral \* 5)

        print()

    elif choose==4:

        x1, y1 = [int(x) for x in input("Координати вершини А чотирикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В чотирикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С чотирикутника АВС: ").split()]

        x4, y4 = [int(x) for x in input("Координати вершини D чотирикутника АВС: ").split()]

        quadrangle = Quadrangle(Point(x1,y1), Point(x2, y2), Point(x3, y3),Point(x4, y4))

        polygons.append(quadrangle)

        print("Ви ввели такий довільний чотирикутник: ")

        print(quadrangle)

        print("Периметер: ",quadrangle.perimeter())

        print("Площа: ", quadrangle.area())

        print("Помножимо на 2 всі координати:")

        print(quadrangle \* 2)

        print()

    elif choose==5:

        x1, y1 = [int(x) for x in input("Координати вершини А пятикутника АВС: ").split()]

        x2, y2 = [int(x) for x in input("Координати вершини В пятикутника АВС: ").split()]

        x3, y3 = [int(x) for x in input("Координати вершини С пятикутника АВС: ").split()]

        x4, y4 = [int(x) for x in input("Координати вершини D пятикутника АВС: ").split()]

        x5, y5 = [int(x) for x in input("Координати вершини E пятикутника АВС: ").split()]

        pentagon = Pentagon(Point(x1,y1), Point(x2, y2), Point(x3, y3),Point(x4, y4), Point(x5, y5))

        polygons.append(pentagon)

        print("Ви ввели такий довільний п'ятикутник: ")

        print(pentagon)

        print("Периметер: ",pentagon.perimeter())

        print("Площа: ", pentagon.area())

        print("Помножимо на 10 всі координати:")

        print(pentagon \* 10)

        print()

    answer = input("Бажаєте продовжити? yes\\no: ")

    if answer.lower()=='n' or answer.lower()=='no':

        exit=True

Робота з модулем shelve:

with shelve.open('test.db') as db:

    triangles=[]

    pentagons=[]

    quadrangles=[]

    id\_triangle = 1

    id\_quadrangle = 1

    id\_pentagon = 1

    for ob in polygons:

        if type(ob)==Triangle or type(ob)==Isosceles or type(ob)==Equilateral:

            js = {

            'id': id\_triangle,

            'angles':ob.angles,

            'type' : str(type(ob).\_\_name\_\_),

            'points': {'A': str(ob.A), 'B': str(ob.B), 'C': str(ob.C)},

            'sides': {'AB':ob.len\_AB, 'AC':ob.len\_AC, 'BC':ob.len\_BC},

            'perimeter':ob.perimeter(),

            'area':ob.area(),

            'details': str(ob)

            }

            if (type(ob)==Isosceles):

                js['height']=ob.heigth()

            triangles.append(js)

            id\_triangle+=1

        elif type(ob)==Quadrangle:

            quadrangles.append( {

            'id': id\_quadrangle,

            'angles':ob.angles,

            'points': {'A': str(ob.A), 'B': str(ob.B), 'C': str(ob.C), 'D':str(ob.D)},

            'sides': {'AB':ob.len\_AB, 'BC':ob.len\_BC, 'CD':ob.len\_CD, 'AD': ob.len\_AD},

            'perimeter':ob.perimeter(),

            'area':ob.area(),

            'details': str(ob) })

            id\_quadrangle+=1

        elif type(ob)==Pentagon:

            pentagons.append( {

            'id': id\_pentagon,

            'angles':ob.angles,

            'points': {'A': str(ob.A), 'B': str(ob.B), 'C': str(ob.C), 'D':str(ob.D), 'E':str(ob.E)},

            'sides': {'AB':ob.len\_AB, 'BC':ob.len\_BC, 'CD':ob.len\_CD, 'DE': ob.len\_DE, 'AE': ob.len\_AE},

            'perimeter':ob.perimeter(),

            'area':ob.area(),

            'details': str(ob) })

            id\_pentagon+=1

    db['Triangles'] = triangles

    db['Quadrangles'] = quadrangles

    db['Pentagons'] = pentagons

with shelve.open('test.db') as db:

    print("\tТрикутники")

    print(db['Triangle'])

    print("\tЧотирикутники")

    print(db['Quadrangle'])

    print("\tП'ятикутники")

    print(db['Pentagons'])

Скріншот результатів:

