Python Tutorial II – Objects and Classes

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 201, Spring Semester 2025

August 29, 2025

Overview

1. Working with Objects and Classes
2. Data Hiding and Encapsulation
3. Relationships Among Classes
4. Inheritance Mechanisms
5. Composition of Object Models
6. Working with Groups of Objects
7. Case Study: GeoModeling Spatial Entities

Composition of Object Models

Composition of Object Models

Definition

Composition is known as is a part of or is a relationship.

The member object is a part of the containing class and the member object cannot survive or exist outside the enclosing or containing class or doesn't have a meaning after the lifetime of the enclosing object.

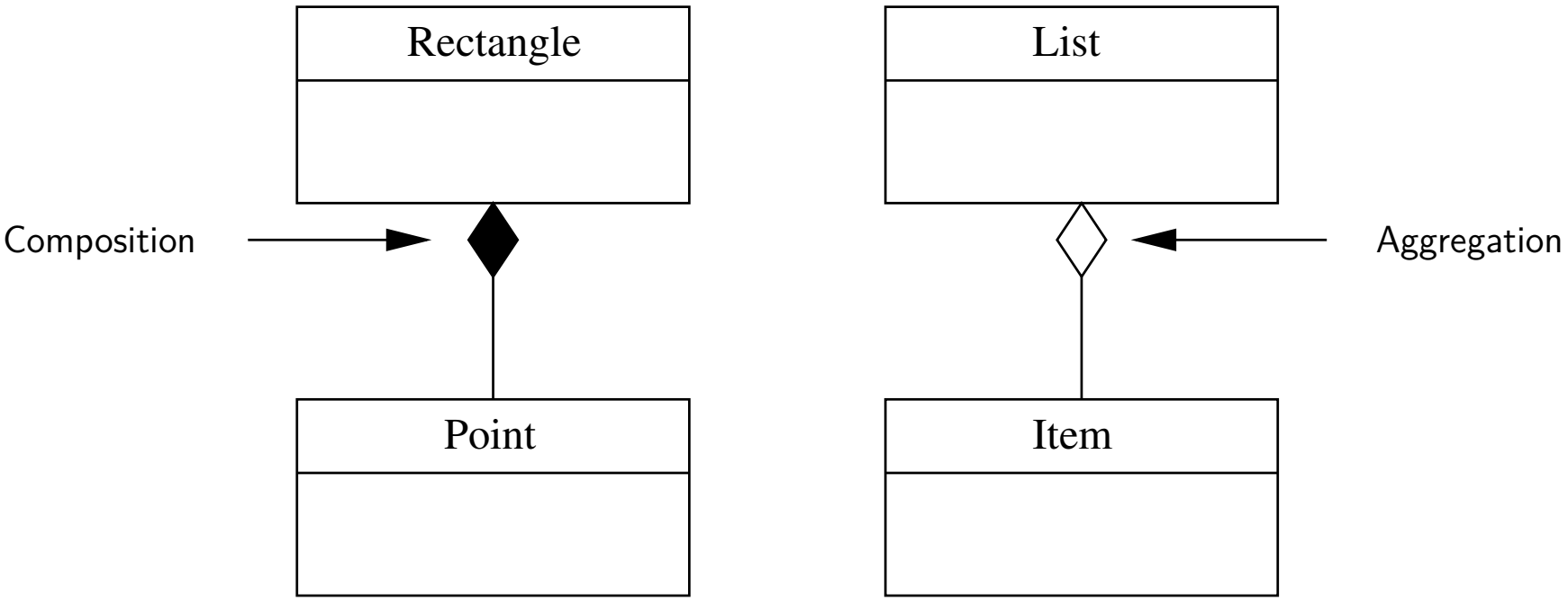
Is it Aggregation or Composition?

- Ask the question: if the part moves, can one deduce that the whole moves with it in normal circumstances?

Example: A car is composition of wheels and an engine. If you drive the car to work, hopefully the wheels go too!

Composition of Object Models

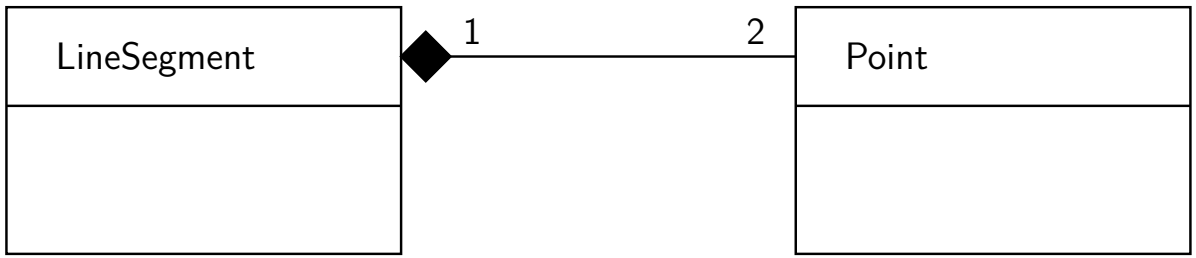
Notation for Aggregation and Composition



Recall: Aggregation is all about grouping of things ...

Example 7. Modeling Line Segments

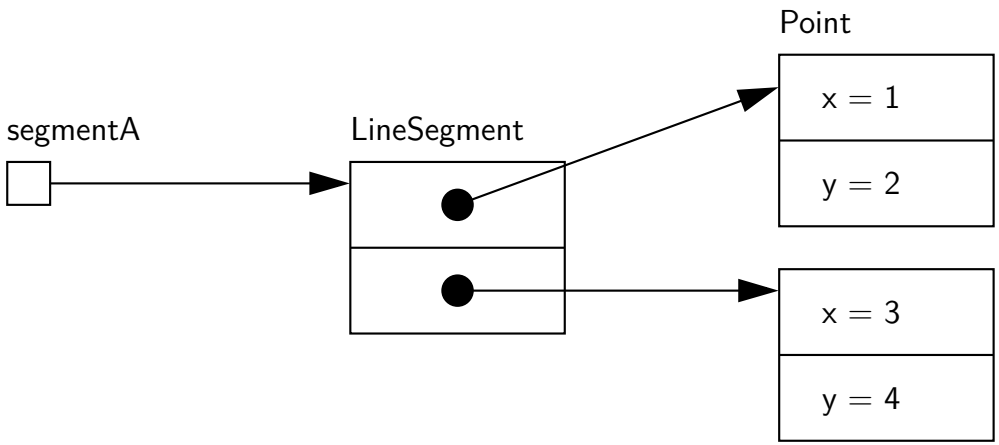
Model Composition



Creating a line segment object with:

```
segmentA = LineSegment( 1, 2, 3, 4 );
```

should give a layout of memory:



Example 7. Modeling Line Segments

Part I: Line Segment Object Model

```
1   # ================================================================
2   # LineSegment.py: Line segments are defined by end points (x1, y1) and
3   # (x2, y2). Compute length and angle of the line segment in radians.
4   #
5   # Written by: Mark Austin                                October, 2022
6   # ================================================================
7
8   import math
9
10  from Point import Point
11
12  class LineSegment:
13    __length = 0
14    __angle  = 0
15
16    def __init__(self, x1, y1, x2, y2 ):
17      self.__pt1 = Point(x1,y1)              # <-- Object composition ...
18      self.__pt2 = Point(x2,y2)              # <-- Object composition ...
19      self.__length = self.__pt1.distance(self.__pt2)
20      self.__angle  = self.getAngle()
21
22    # Compute angle (radians) for coordinates in four quadrants ....
23
24    def getAngle(self):
25       dX = self.__pt2.get_xCoord() - self.__pt1.get_xCoord();
26       dY = self.__pt2.get_yCoord() - self.__pt1.get_yCoord();
```

Example 7. Modeling Line Segments

Part I: Line Segment Object Model (Continued) ...

```
27
28          if dY >  0.0 and dX == 0.0:
29              angle = math.pi/2.0
30          if dY >= 0.0 and dX > 0.0:
31              angle = math.atan( dY/dX )
32          if dY >= 0.0 and dX < 0.0:
33              angle = math.pi + math.atan( dY/dX )
34          if dY < 0.0 and dX < 0.0:
35              angle = math.pi + math.atan( dY/dX )
36          if dY < 0.0 and dX >= 0.0:
37              angle = 2*math.pi + math.atan( dY/dX )
38
39          return angle
40
41      # String represention of line segment ...
42
43      def __str__(self):
44          x1 = self.__pt1.get_xCoord();
45          y1 = self.__pt1.get_yCoord();
46          x2 = self.__pt2.get_xCoord();
47          y2 = self.__pt2.get_yCoord();
48          return "--- LineSegment: (x1,y1) = (%5.2f, %5.2f), (x2,y2) = (%5.2f, %5.2f),
49                      angle = %.2f, length = %.2f" % ( x1, y1, x2, y2, self.__angle, self.__l
```

Example 7. Modeling Line Segments

Part II: Line Segment Test Program

```
1  # ======================================================
2  # TestLineSegment.py: Exercise line segment class ...
3  # ======================================================
4
5  from LineSegment import LineSegment
6
7  # main method ...
8
9  def main():
10     print("--- Enter TestLineSegment.main()    ... ");
11     print("--- ============================== ... ");
12
13     print("--- Part 1: Create test line segment ... ");
14
15     segmentA = LineSegment( 1.0, 2.0,  3.0,  4.0 )
16     print(segmentA)
17
18     print("--- Part 2: Sequence of line segments ... ");
19
20     a = LineSegment( 0.0, 0.0,  3.0,  0.0 )
21     print(a)
22     b = LineSegment( 0.0, 0.0,  3.0,  3.0 )
23     print(b)
24     c = LineSegment( 0.0, 0.0,  0.0,  3.0 )
25     print(c)
26     d = LineSegment( 0.0, 0.0, -3.0,  3.0 )
27     print(d)
```

Example 7. Modeling Line Segments

Part II: Line Segment Test Program (Continued) ...

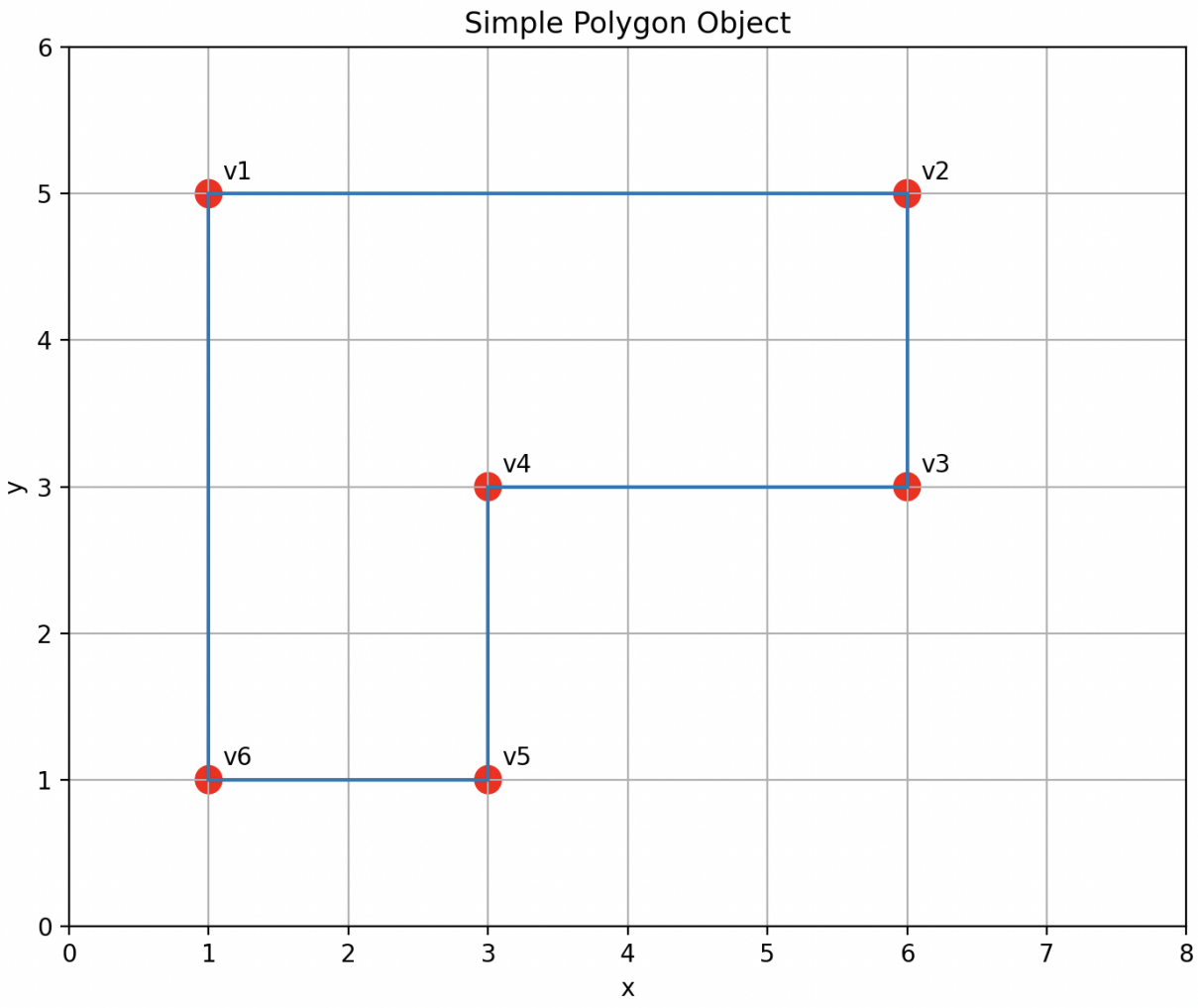
```
28      e = LineSegment( 0.0, 0.0, -3.0,  0.0 )
29      print(e)
30
31      print("--- ============================== ... ");
32      print("--- Finished TestLineSegment.main() ... ");
33
34  # call the main method ...
35
36  main()
```

Part III: Abbreviated Program Output:

```
--- Part 1: Create test line segment ...
--- LineSegment: (x1,y1) = ( 1.00,  2.00), (x2,y2) = ( 3.00,  4.00), angle = 0.79, length = 2.83
--- Part 2: Sequence of line segments ...
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = ( 3.00,  0.00), angle = 0.00, length = 3.00
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = ( 3.00,  3.00), angle = 0.79, length = 4.24
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = ( 0.00,  3.00), angle = 1.57, length = 3.00
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = (-3.00,  3.00), angle = 2.36, length = 4.24
--- LineSegment: (x1,y1) = ( 0.00,  0.00), (x2,y2) = (-3.00,  0.00), angle = 3.14, length = 3.00
```

Source Code: See: python-code.d/classes/

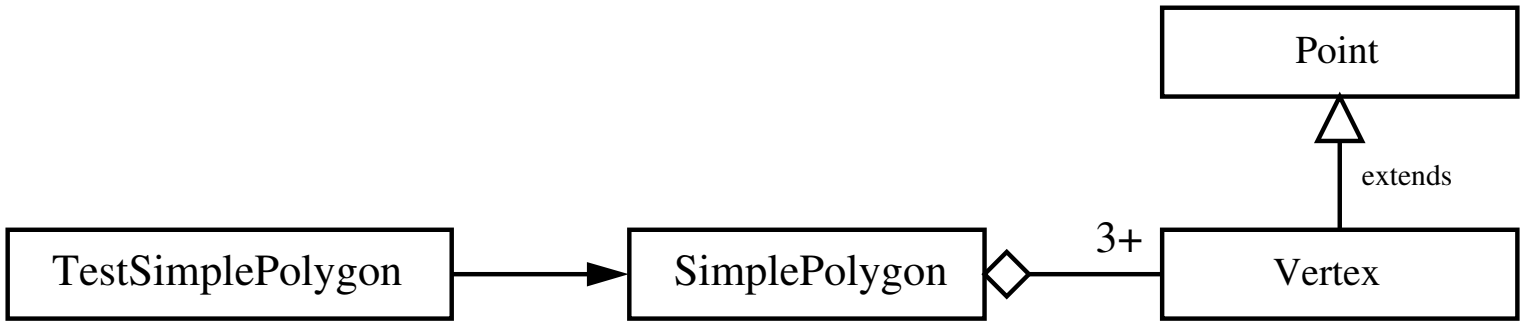
Example 8. Simple Polygon Object



Source Code: See: python-code.d/applications/shapes/TestSimplePolygon01.py

Example 8. Simple Polygon Object

Part I: Program Architecture



Point Attributes:

- `_xCoord`, `_yCoord`, ...

Vertex Attributes:

- `_label` ...

SimplePolygon Attributes:

- `v01`, `v02`, ... `v06`, `polygon01`, ...

Example 8. Simple Polygon Object

Part II: Abbreviated Program Output:

```
--- Enter TestSimplePolygon01.main()    ...
--- =================================== ...

--- Part 1: Create list of vertices ...

--- Part 2: Assemble and print simple polygon object ...

--- SimplePolygon: L-shaped Polygon ...
--- ------------------------------------------------
---   Vertex  1: (x,y) = (  1.00,   5.00) ...
---   Vertex  2: (x,y) = (  6.00,   5.00) ...
---   Vertex  3: (x,y) = (  6.00,   3.00) ...
---   Vertex  4: (x,y) = (  3.00,   3.00) ...
---   Vertex  5: (x,y) = (  3.00,   1.00) ...
---   Vertex  6: (x,y) = (  1.00,   1.00) ...
--- Perimeter    =  18.00 ...
--- ------------------------------------------------
--- Part 3: Draw simple polygon ...

--- ==================================== ...
--- Finished TestSimplePolygon01.main()     ...
```

Example 8. Simple Polygon Object

Part III: Point Object Model

```
1   # ============================================================
2   # Point01.py: Bare-bones implementation of a Point class ...
3   #
4   # Written by: Mark Austin                                October, 2024
5   # ============================================================
6
7   import math
8
9   class Point:
10
11      def __init__(self, xCoord=0, yCoord=0):
12          self._xCoord = xCoord
13          self._yCoord = yCoord
14
15      # Get/set X coordinate
16
17      def getX(self):
18          return self._xCoord
19
20      def setX(self, xCoord):
21          self._xCoord = xCoord
22
23      # Get/set Y coordinate
24
25      def getY(self):
26          return self._yCoord
```

Example 8. Simple Polygon Object

Part III: Point Object Model (Continued) ...

```
27
28      def setY(self, yCoord):
29          self._yCoord = yCoord
30
31      # Get current position
32
33      def get_position(self):
34          return self.getX(), self.getY()
35
36      # Change x & y coordinates by p & q
37
38      def move(self, p, q):
39          self._xCoord += p
40          self._yCoord += q
41
42      # Compute distance between two points ...
43
44      def distance(self, second):
45          x_d = self.getX() - second.getX()
46          y_d = self.getY() - second.getY()
47          return (x_d**2 + y_d**2)**0.5
48
49      # Return string represention of object ...
50
51      def __str__(self):
52          return "( %6.2f, %6.2f )" % ( self.getX(), self.getY() )
```

Example 8. Simple Polygon Object

Part III: Vertex Object Model

```
1   # ============================================================
2   # Vertex.py: A vertex is a point with a label ...
3   # ============================================================
4
5   from Point01 import Point
6
7   class Vertex(Point):
8       _label = ""
9
10      # Constructor method ...
11
12      def __init__(self, x, y) :
13        Point.__init__(self, x, y)
14        self._label = ""
15
16      # Set/get label ...
17
18      def setLabel(self, label ):
19        self._label = label
20
21      def getLabel(self):
22        return self._label
```

Example 8. Simple Polygon Object

Part III: Vertex Object Model (Continued) ...

```
23
24      # Assemble string represention of Vertex ...
25
26      def __str__(self):
27          vertexinfo = [];
28          vertexinfo.append("\n");
29          vertexinfo.append("--- Vertex: {:s} ... \n".format( self.getLabel()));
30          vertexinfo.append("--- ------------------------------------------------ \n");
31          vertexinfo.append("---   Coordinate: (x,y) = {:s} ... \n".format( Point.__self__()
32          vertexinfo.append("--- ------------------------------------------------ ");
33          return "".join(vertexinfo);
```

Example 8. Simple Polygon Object

Part IV: Simple Polygon Object Model

```
1   # ====================================================================
2   # SimplePolygon01.py: Bare-bones implementation of a simple polygon.
3   #
4   # Written by: Mark Austin                                  October 2024
5   # ====================================================================
6
7   import math
8
9   from Vertex01 import Vertex
10  from matplotlib.patches import Circle
11  from matplotlib.lines import Line2D
12
13  class SimplePolygon:
14    area      = 0
15    perimeter = 0
16    name      = ""
17    coords = []
18
19    # Constructor method ...
20
21    def __init__(self, vertexlist):
22      self.coords    = vertexlist;          # <--- Assign vertex list to coords ...
23      self.perimeter = self.getPerimeter()  # <--- Compute perimeter ...
24
25    # Set/get name ...
26
27    def setName(self, name):
```

Example 8. Simple Polygon Object

Part IV: Simple Polygon Object Model (Continued) ...

```
28        self.name = name
29
30      def getName(self):
31        return self.name
32
33      # Compute polygon perimeter ...
34
35      def getPerimeter(self):
36
37         dperimeter = 0.0;
38         for i in range( len(self.coords)-1):
39             dperimeter += self.coords[i].distance( self.coords[i+1] );
40
41         lastnode = len( self.coords) - 1
42         dperimeter += self.coords[ lastnode ].distance( self.coords[0] );
43
44         return dperimeter;
45
46      # Draw simple polygon ...
47
48      def draw(self, ax):
49
50          # Draw polygon edges ...
51
52          for i in range( len(self.coords)-1):
53              xcoords = [ self.coords[i].getX(), self.coords[i+1].getX() ];
54              ycoords = [ self.coords[i].getY(), self.coords[i+1].getY() ];
55              ax.add_line( Line2D(xcoords, ycoords) )
```

Example 8. Simple Polygon Object

Part IV: Simple Polygon Object Model (Continued) ...

```
56
57          lastnode = len( self.coords) - 1
58          xcoords = [ self.coords[0].getX(), self.coords[ lastnode ].getX() ];
59          ycoords = [ self.coords[0].getY(), self.coords[ lastnode ].getY() ];
60          ax.add_line( Line2D(xcoords, ycoords) )
61
62          # Draw polygon vertices as small circles ...
63
64          width = 0.1;
65          for i in range(len( self.coords )):
66              xcoord = self.coords[i].getX();
67              ycoord = self.coords[i].getY();
68              ax.add_patch( Circle( (xcoord, ycoord), width, facecolor='red') )
69
70          # Draw node labels ...
71
72          dx = 0.1; dy = 0.1
73          for i in range(len( self.coords )):
74              xcoord = self.coords[i].getX();
75              ycoord = self.coords[i].getY();
76              label  = self.coords[i].getLabel();
77              ax.text( xcoord + dx, ycoord + dy, label )
```

Example 8. Simple Polygon Object

Part IV: Simple Polygon Object Model (Continued) ...

```
78
79      # String represention of simple polygon ...
80
81      def __str__(self):
82          polygoninfo = [];
83          polygoninfo.append("\n");
84          polygoninfo.append("--- SimplePolygon: {:s} ...  \n".format( self.name ));
85          polygoninfo.append("--- ------------------------------------------------ \n");
86
87          for i in range(len( self.coords )):
88              xc = self.coords[i].getX();
89              yc = self.coords[i].getY();
90              polygoninfo.append("---   Vertex {:2d}: (x,y) = ({:6.2f}, {:6.2f}) ... \n".for
91
92          polygoninfo.append("--- Perimeter    = {:6.2f} ... \n".format( self.getPerimeter()
93          polygoninfo.append("--- ------------------------------------------------ ");
94          return "".join(polygoninfo);
```

Example 8. Simple Polygon Object

Part V: Simple Polygon Test Program

```
1   # ============================================================
2   # TestSimplePolygon01.py: Exercise SimplePolygon class ...
3   # ============================================================
4
5   from Vertex01 import Vertex
6   from SimplePolygon01 import SimplePolygon
7
8   import matplotlib.pyplot as plt
9
10  # main method ...
11
12  def main():
13      print("--- Enter TestSimplePolygon01.main()     ... ");
14      print("--- ==================================== ... ");
15
16      print("--- Part 1: Create list of vertices ... ");
17
18      v01 = Vertex ( 1.0, 5.0 ); v01.setLabel("v1");
19      v02 = Vertex ( 6.0, 5.0 ); v02.setLabel("v2");
20      v03 = Vertex ( 6.0, 3.0 ); v03.setLabel("v3");
21      v04 = Vertex ( 3.0, 3.0 ); v04.setLabel("v4");
22      v05 = Vertex ( 3.0, 1.0 ); v05.setLabel("v5");
23      v06 = Vertex ( 1.0, 1.0 ); v06.setLabel("v6");
24
25      print("--- Part 2: Assemble and print simple polygon object ... ");
```

Example 8. Simple Polygon Object

Part V: Simple Polygon Test Program (Continued) ...

```
27        polygon01 = SimplePolygon( [ v01, v02, v03, v04, v05, v06 ] )
28        polygon01.setName("L-shaped Polygon")
29
30        print( polygon01 )
31
32        print("--- Part 3: Draw simple polygon ... \n");
33
34        # Define Matplotlib figure and axis
35
36        fig, ax = plt.subplots()
37
38        polygon01.draw(ax)
39
40        plt.title('Simple Polygon Object')
41        plt.ylabel('y')
42        plt.xlabel('x')
43        plt.ylim( 0, 6 )
44        plt.xlim( 0, 8 )
45        plt.grid(True)
46        plt.show()
47
48        print("--- ======================================== ... ");
49        print("--- Finished TestSimplePolygon01.main()     ... ");
50
51    # call the main method ...
52
53    main()
```