# Python Tutorial – Part 2: Objects and Classes

Mark A. Austin

University of Maryland

*austin@umd.edu*
*ENCE 688P, Spring Semester 2022*

February 20, 2023

# Overview



Part 1

# Working with Objects and Classes

# Working with Objects and Classes

**Working with Objects and Classes:**

- Collections of objects share similar traits (e.g., data, structure, behavior).
- Collections of objects will form relationships with other collections of objects.

**Definition of a Class**

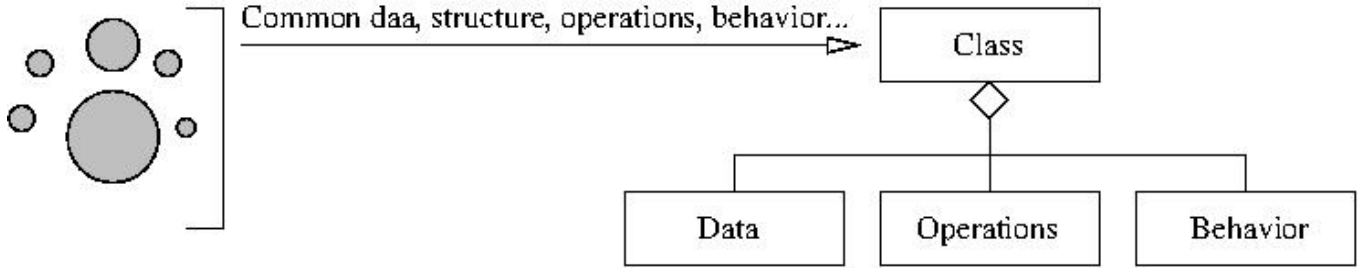
A class is a specification (or blueprint) of an object's structure and behavior.

**Definition of an Object**

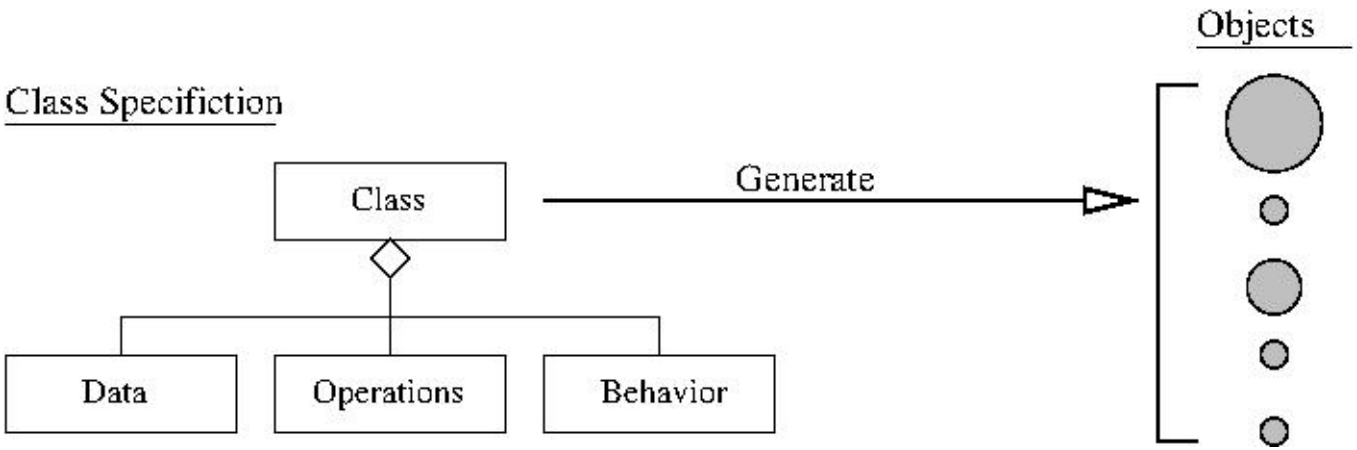
An object is an instance of a class.

# Working with Objects and Classes

## From Collections of Objects to Classes:




## Generation of Objects from Class Specifications:



# Working with Objects and Classes

**Principles for Development of Reusable Code:**

- **Inheritance:** Create new (specialized) classes from existing classes through mechanism of concept extension.
- **Encapsulation:** Hide some details of a class from other (external) classes.
- **Polymorphism:** Use common operation in different ways depending on details of data input.

**Key Design Tasks**

- Identify objects and their attributes and functions,
- Establish relationships among the objects,
- Implement and test the individual objects,
- Assemble and test the system.

# Example 1. Working with Points

## A Very Simple Class in Python

```
1   # ========================================================
2   # Point.py: Create point objects ...
3   #
4   # Modified by: Mark Austin                        October, 2020
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      # compute distance between two points ...
16
17      def distance(self, second):
18          x_d = self.__xCoord - second.__xCoord
19          y_d = self.__yCoord - second.__yCoord
20          return (x_d**2 + y_d**2)**0.5
21
22      # return string represention of object ...
23
24      def __str__(self):
25          return "( %6.2f, %6.2f ) " % ( self.__xCoord, self.__yCoord )
```

# Example 1. Working with Points

## Create and Print two Point Objects

```
 8       pt1 = Point( 0.0, 0.0 )
 9       pt2 = Point( 3.0, 4.0 )
10
11       print("--- pt1 = %s ..." % (pt1) )
12       print("--- pt2 = %s ..." % (pt2) )
```

## Output:

```
--- pt1 = (   0.00,   0.00 )  ...
--- pt2 = (   3.00,   4.00 )  ...
```

## Compute Distance between Two Points

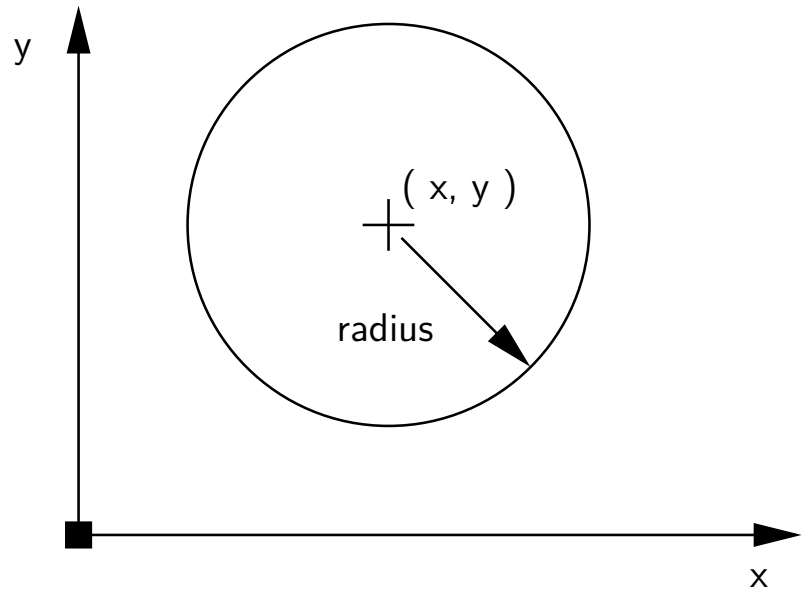
```
10       distance = pt1.distance(pt2)
11       print("--- Distance between pt1 and pt2 --> %.2f ..." % (distance) )
```

## Output:

```
--- Distance between pt1 and pt2 --> 5.00 ...
```

# Example 2. Working with Circles

A circle can be described by the `(x,y)` position of its center and by its `radius`.




There are numerous things we can do with circles:

- Compute their circumference, perimeter or area,
- Check if a point is inside a circle.

# Example 2. Working with Circles

```
1   # ========================================================
2   # Circle.py: Simplified modeling of a circle ...
3   #
4   # Written by: Mark Austin                        October, 2020
5   # ========================================================
6
7   import math
8
9   class Circle:
10    radius = 0
11    area   = 0
12    perimeter = 0
13
14    def __init__(self, x, y, radius):
15      self.radius    = radius
16      self.area      = math.pi*radius*radius
17      self.perimeter = 2.0*math.pi*radius
18      self.x = x
19      self.y = y
20
21    # Set circle radius, recompute area and perimeter ...
22
23    def setRadius(self, radius):
24      self.radius = radius
25      self.area   = math.pi*radius*radius
26      self.perimeter = 2.0*math.pi*radius
```

# Example 2. Working with Circles

```
27
28      # Print details of circle ...
29
30      def printCircle(self):
31        print("--- Circle: (x,y) = (%.2f, %.2f): radius = %.2f: area = %.2f: perimeter = %.2
32              % ( self.x, self.y, self.radius, self.area, self.perimeter ) )
```

## Create and Print two Circle Objects

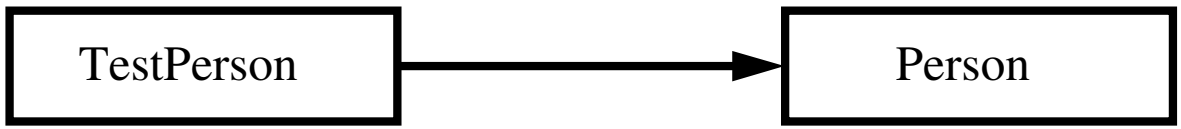
```
1       x = Circle( 0.0, 0.0, 3.0 )
2       y = Circle( 1.0, 2.0, 4.0 )
3       x.printCircle()
4       y.printCircle()
```

## Output:

```
--- Circle: (x,y) = (0.00, 0.00): radius = 3.00: area = 28.27
--- Circle: (x,y) = (1.00, 2.00): radius = 4.00: area = 50.27
```

# Example 3. Object Model of a Person

**Part I: Program Architecture.** The TestPerson will create objects of type Person.




## Part II: Person Object Model:

```
1   # ========================================================
2   # Person.py: Simplified model of a person ...
3   #
4   # Written by: Mark Austin                        October, 2022
5   # ========================================================
6
7   class Person:
8     age = 0
9     ssn = 0
10
11    def __init__(self, fname, lname):
12      self.firstname = fname
13      self.lastname  = lname
14
15    def printname(self):
16      print("--- Name: {:s}, {:s}".format( self.firstname, self.lastname) )
```

# Example 3. Object Model of a Person

## Part II: Person Object Model: (Continued) ...

```
17
18      # Get first and last names ...
19
20      def getFirstName(self):
21        return self.firstname
22
23      def getLastName(self):
24        return self.lastname
25
26      # Set/print age ...
27
28      def setAge(self, age):
29        self.age = age
30
31      def printAge(self):
32        print("--- Age = {:d} ".format(self.age) )
33
34      # Set/print social security number ...
35
36      def setSSN(self, ssn ):
37        self.ssn = ssn
38
39      def printSSN(self):
40        print("--- Social Security No: {:d} ...".format(self.ssn) )
```

# Example 3. Object Model of a Person

## Part III: Person Test Program:

```
1  # ======================================================
2  # TestPerson.py: Test program for person objects ...
3  # ======================================================
4
5  from Person import Person
6
7  # main method ...
8
9  def main():
10     print("--- Enter TestPerson.main()          ... ");
11     print("--- ================================ ... ");
12
13     # Exercise methods in class Person ...
14
15     x = Person( "Angela", "Austin" )
16     x.printname()
17
18     print("--- First name: {:s} ".format( x.getFirstName() ) )
19     print("--- Family name: {:s} ".format( x.getLastName() ) )
20
21     # Initialize attribute values ..
22
23     x.setAge(29)
24     x.setSSN(123456789)
25
26     # Print attribute values ..
```

# Example 3. Test Program for Person Object Model

## Part III: Person Test Program: (Continued) ...

```
28      x.printAge()
29      x.printSSN()
30
31      print("--- ============================== ... ");
32      print("--- Finished TestPerson.main()      ... ");
33
34  # call the main method ...
35
36  main()
```

## Output:

```
--- Enter TestPerson.main()          ...
--- ============================== ...
--- Name: Angela, Austin
--- First name: Angela
--- Family name: Austin
--- Age = 29
--- Social Security No: 123456789
--- ============================== ...
--- Finished TestPerson.main()       ...
```

# Example 3. Object Model of a Person

## Part IV: Files before Program Execution:

```
-rw-r--r--  1 austin  staff   903 Feb 18 13:21 Person.py
-rw-r--r--  1 austin  staff   847 Feb 18 13:26 TestPerson.py
```

## Part IV: Files after Program Execution:

```
-rw-r--r--  1 austin  staff   903 Feb 18 13:21 Person.py
-rw-r--r--  1 austin  staff   847 Feb 18 13:26 TestPerson.py
drwxr-xr-x  4 austin  staff   128 Feb 18 13:27 __pycache__

./__pycache__:
total 16
-rw-r--r--  1 austin  staff  1476 Feb 18 13:27 Person.cpython-37.pyc
```

**Note:** When TestPerson imports Person, python builds a compiled bytecode for Person (with .pyc extension).

Subsequent imports will be easier and faster.

# Data Hiding and Encapsulation

# Hiding Information

## Data Hiding

Data Hiding is isolation of the client from a part of program implementation. Some objects in the module are kept internal, invisible, and inaccessible to the user.

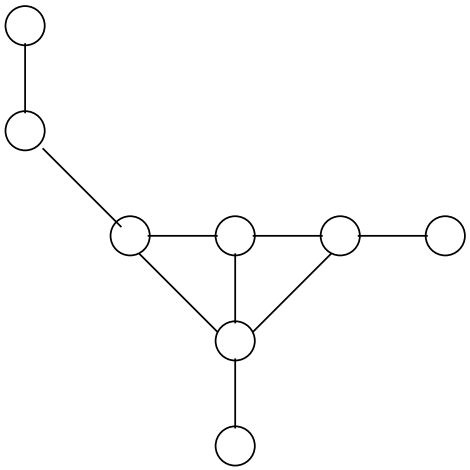
## Principle of Information Hiding

The principle of information hiding states that information which is likely to change (e.g., over the lifetime of a software/systems package) should be hidden inside a module.

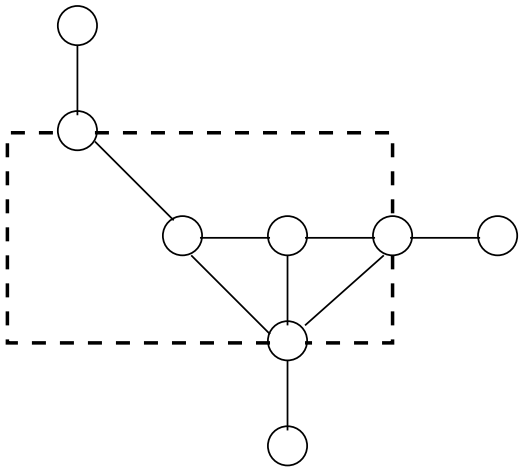
**Key Advantages**

- Prevents accidental linkage to incorrect data.
- It heightens the security against hackers that are unable to access confidential data.

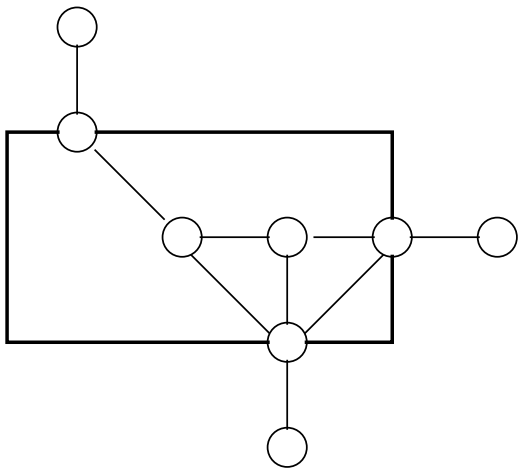
# Data Hiding and Encapsulation



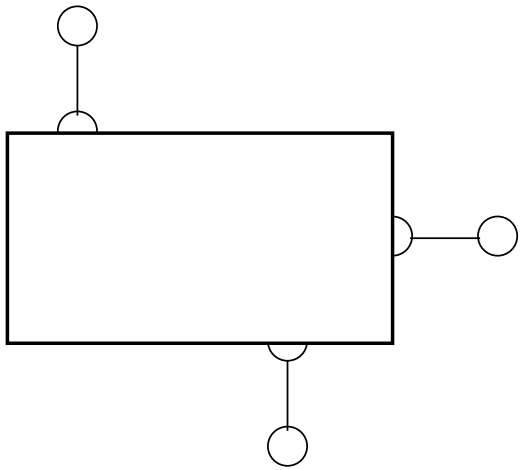
Unstructured Components



Aggregation



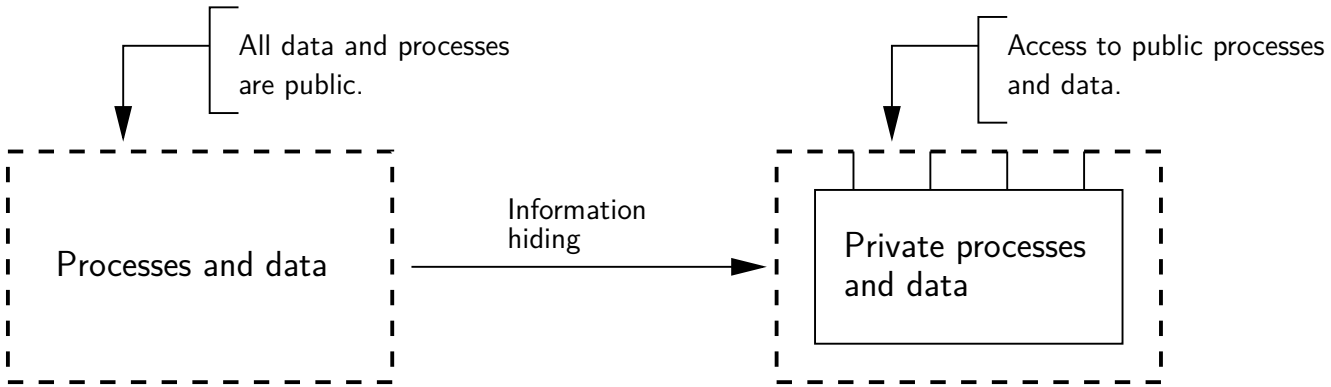
Designer's view of Aggregation



Encapsulation – User's view of Abstraction

# Data Hiding and Encapsulation

**Application.** Process for Implementation of Information Hiding.




**Data Hiding in Python** (Private and Protected) ...

- Data hiding is implemented by using a double underscore before (prefix) the attribute name. Making an attribute private hides it from users.
- Use of a single underscore makes the variable/method protected. The variables/methods will be available to the class, and all of its subclasses.

# Example 4. Revised Circle Object Model

## Part I: Revised Circle Object Model

```
1   # ============================================================
2   # Circle.py: Implementation of circle model with encapsulation
3   # (hiding) of circle parameters and properties.
4   #
5   # Written by: Mark Austin                              October, 2020
6   # ============================================================
7
8   import math
9
10  class Circle:
11    __radius = 0            # <-- private parameters ....
12    __area   = 0
13    __perimeter = 0
14
15    def __init__(self, x, y, radius):
16      self.__radius    = radius
17      self.__area      = math.pi*radius*radius
18      self.__perimeter = 2.0*math.pi*radius
19      self.__x = x
20      self.__y = y
21
22    # Set circle coordinates ...
23
24    def setX(self, x):
25      self.__x = x
```

# Example 4. Revised Circle Object Model

## Part I: Revised Circle Object Model (Continued) ...

```
27      def setY(self, y):
28        self.__y = y
29
30      # Set circle radius, recompute area and perimeter ...
31
32      def setRadius(self, radius):
33        self.__radius = radius
34        self.__area   = math.pi*radius*radius
35        self.__perimeter = 2.0*math.pi*radius
36
37      # Get circle parameters ...
38
39      def getX(self):
40        return self.__x
41
42      def getY(self):
43        return self.__y
44
45      def getRadius(self):
46        return self.__radius
47
48      def getArea(self):
49        return self.__area
50
51      def getPerimeter(self):
52        return self.__perimeter
```

# Example 4. Revised Circle Object Model

## Part I: Revised Circle Object Model (Continued) ...

```
54      # String represention of circle ...
55
56      def __str__(self):
57          return "--- Circle: (x,y) = (%.2f, %.2f): radius = %.2f: area = %.2f:
58                  perimeter = %.2f" % ( self.__x, self.__y, self.__radius,
59                  self.__area, self.__perimeter )
```

## Part II: Test Program for Circle Object Model

```
1   # ========================================================
2   # TestCircles.py: Exercise circle objects.
3   #
4   # Written by: Mark Austin                          December 2022
5   # ========================================================
6
7   from Circle import Circle
8
9   # main method ...
10
11  def main():
12      print("--- Enter TestCircles.main()          ... ");
13      print("--- ================================ ... ");
14
15      print("--- Part 1: Create and print circle ... ");
16
17      x = Circle( 0.0, 0.0, 3.0 )
18      print(x)
```

# Example 4. Revised Circle Object Model

## Part II: Test Program for Circle Object Model (Continued) ...

```
20      print("--- ============================== ... ");
21      print("--- Finished TestCircles.main()     ... ");
22
23  # call the main method ...
24
25  main()
```

## Part III: Program Output

```
--- Enter TestCircles.main()          ...
--- ============================== ...
--- Circle: (x,y) = (0.00, 0.00): radius = 3.00: area = 28.27
--- ============================== ...
--- Finished TestCircles.main()     ...
```