Object-Oriented Approach in Python Programming

Presented By

Dr. Srinivas Narasegouda,

Assistant Professor,

Jyoti Nivas College Autonomous,

Bangalore - 95

Object-Oriented Approach in Python cont.

Object-Oriented Approach: The Object-Oriented Approach, **Custom Classes:** Attributes and Methods, Inheritance and Polymorphism, Using Properties to Control Attribute Access.

Exceptions and tools: Default Exception Handler, Catching Exceptions, Raising Exceptions, User-Defined Exceptions, Termination Actions.

Object-Oriented Approach in Python cont.

The Object-Oriented Approach :

- One of the popular approaches to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).
- An object has two characteristics:
  1. Attributes
  2. Behavior
- A parrot can be an object, as it has the following properties:
  - name, age, color as attributes
  - singing, dancing as behavior
- The concept of OOP in Python focuses on creating reusable code. This concept is also known as **DRY (Don't Repeat Yourself)**.

Object-Oriented Approach in Python cont.

Object-Oriented vs Procedural :

Sl. No	Object-oriented Programming	Procedural Programming
1.	Object-oriented programming is the problem-solving approach and used where computation is done by using objects.	Procedural programming uses a list of instructions to do computation step by step.
2.	It makes the development and maintenance easier.	In procedural programming, It is not easy to maintain the codes when the project becomes lengthy.
3.	It simulates the real world entity. So real-world problems can be easily solved through oops.	It doesn't simulate the real world. It works on step by step instructions divided into small parts called functions.
4.	It provides data hiding. So it is more secure than procedural languages. You cannot access private data from anywhere.	Procedural language doesn't provide any proper way for data binding, so it is less secure.

Object-Oriented Approach in Python cont.

Custom Classes:

- **Class** – A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

How to Define a Class?

- All class definitions start with the class keyword, which is followed by the name of the class and a colon. Any code that is indented below the class definition is considered part of the class's body.

```
class class_name:
    #Code_1
    #Code_2
#Code_3 Note: Not part of the class
```

Object-Oriented Approach in Python cont.

Object

- An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.

Object-Oriented Approach in Python cont.

Custom Functions

Attributes and Methods

- Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.
- A method in python is **somewhat similar to a function, except it is associated with object/classes**. Methods in python are very similar to functions except for two major differences.
  1. The method is implicitly used for an object for which it is called.
  2. The method is accessible to data that is contained within the class.

General Syntax:

```
class ClassName:
  def method_name( ):
    …………..
    # Method_body
    ………………
```

- ****Note**** Before writing any function, check if that function is already written or included in the python

Object-Oriented Approach in Python cont.

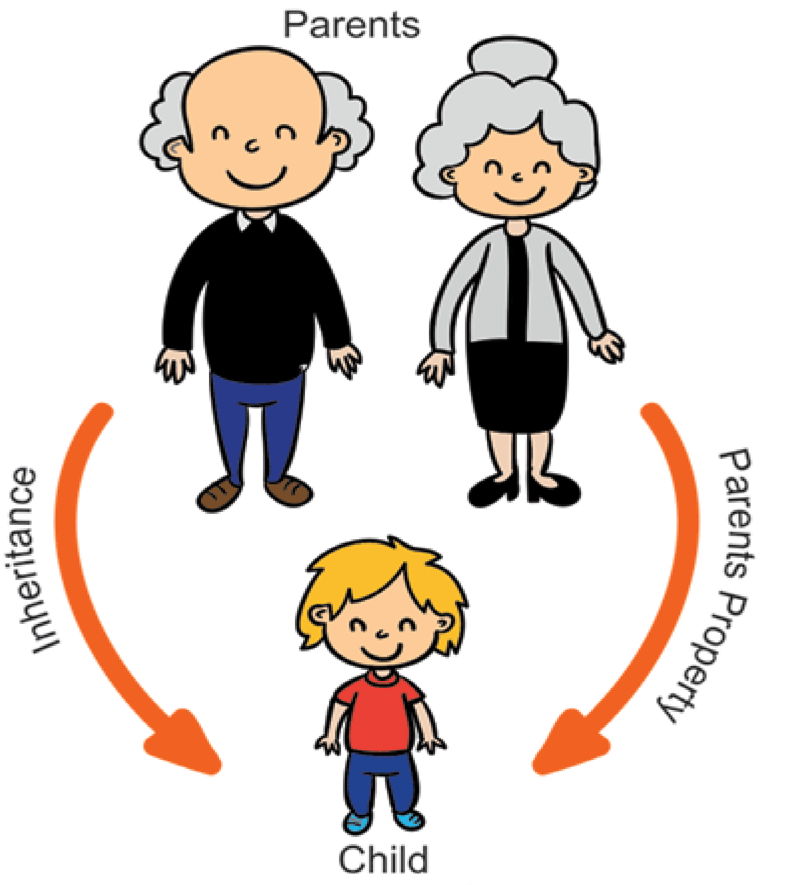
```
class car:
    def __init__(self, modelname, year):
        self.modelname = modelname
        self.year = year
    def display(self):
        print(self.modelname,self.year)
c1 = car("Nexon", 2016) # Object creation
c1.display( )
```

- **__init__( )** any number of parameters, but the first parameter will always be a variable called self.
- When a new class instance is created, the instance is automatically passed to the self parameter in . **__init__( )** so that new attributes can be defined on the object.

Object-Oriented Programming cont.

Inheritance

YOUR MUM AND I ARE THINKING ABOUT GIVING YOU POWER OF ATTORNEY...
PETER F. ROY
YES! YES! YES! I'M RICH! RICH!
...AND NOW, STRANGELY, I'M HAVING SECOND THOUGHTS.
CARTOONSTOCK
Search ID: pron273



- Inheritance is a way of creating a new class for using details of an existing class without modifying it.
- The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).

Object-Oriented Approach in Python cont.

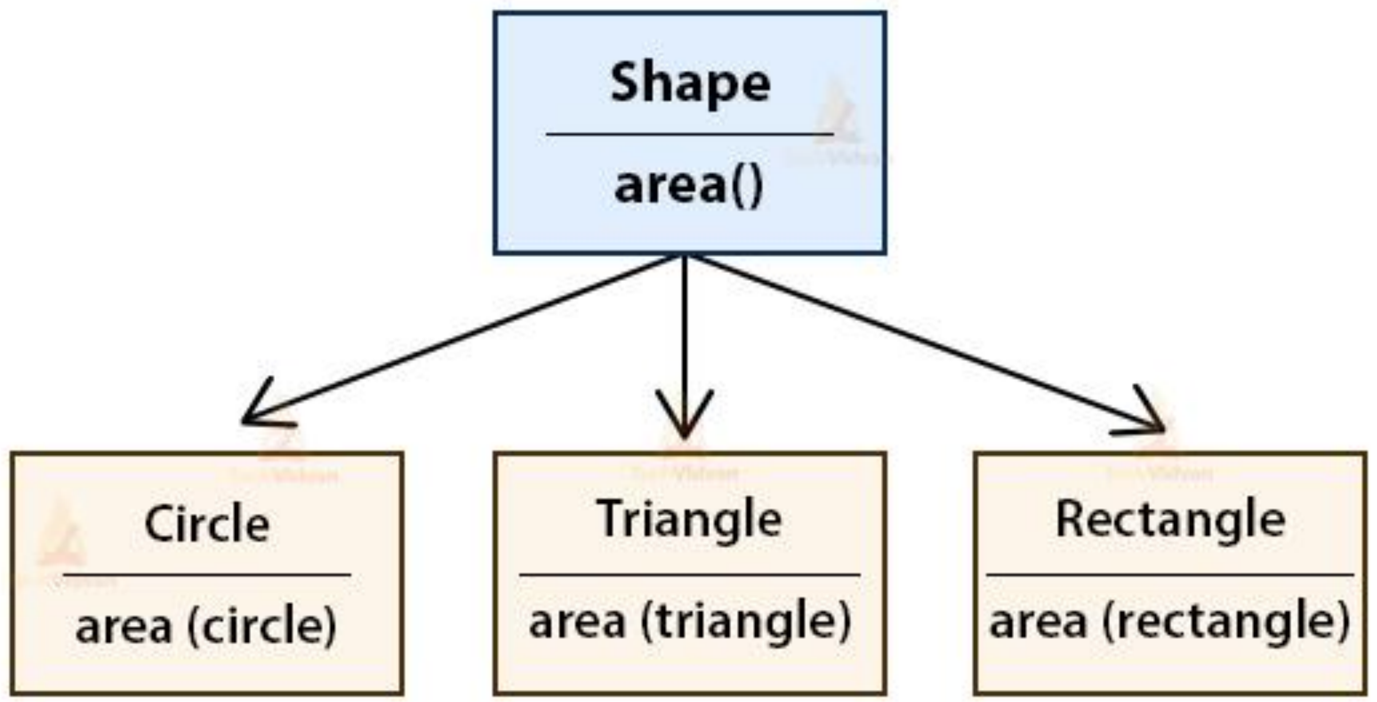
Encapsulation

- Using OOP in Python, we can restrict access to methods and variables.
- This prevents data from direct modification which is called encapsulation.
- In Python, we denote private attributes using underscore as the prefix i.e single _ or double __.

Object-Oriented Approach in Python cont.

Polymorphism

- Polymorphism is an ability (in OOP) to use a common interface for multiple forms (data types).



Access Specifiers & Data Abstraction

What is Data Abstraction?

Data Abstraction in Python is the process of hiding the real implementation of an application from the user and emphasizing only on usage of it.

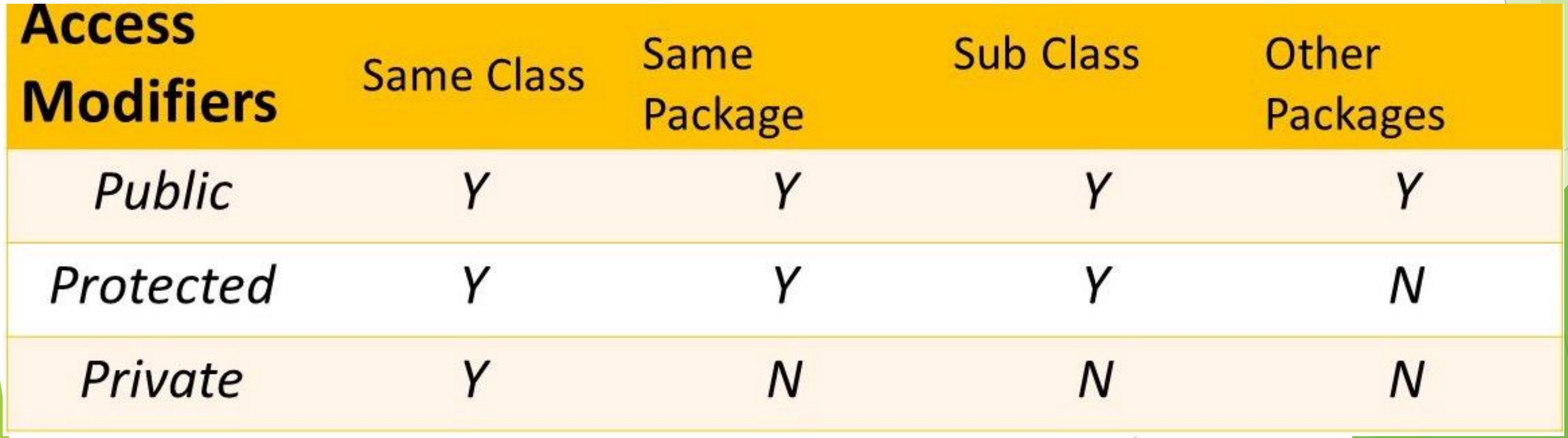
- E.g. when you use mobile, we need not know how pressing a key changes the volume level. We just need to know that pressing up button increases the volume and the down button reduces the volume.

Why Do We Need Abstraction?

Through the process of abstraction in Python, a programmer can hide all the irrelevant data/process of an application in order to reduce complexity, provide security and increase the efficiency.

How is Abstraction achieved?

- Abstraction is achieved with the help of access specifiers.



Access Modifiers	Same Class	Same Package	Sub Class	Other Packages
Public	*Y*	*Y*	*Y*	*Y*
Protected	*Y*	*Y*	*Y*	*N*
Private	*Y*	*N*	*N*	*N*

Object-Oriented Approach in Python cont.

Using Properties to Control Attribute Access.

```
class Mydata:

     # constructor
     def __init__(self, name, gender, age ):

           # public data mambers
           self.myName = name
           self._myGender = gender
           self.__myAge = age

     # public memeber function
     def displayData(self):

           # accessing public data member
           print(“Name: ", self.myName)
           # accessing protected data member
           print(“Gender: ", self._myGender)
           # accessing private data member
           print(“Age: ", self.__myAge)


# creating object of the class
obj = Mydata("Rani", “Female", 21)


# calling public member function of the class
obj.displayData()
# accessing public data member
print("Name: ", obj.myName)
# accessing protected data member
print(“Gender: ", obj.myGender)
# accessing private data member
print(“Age: ", obj.myAge)
```

```
OUTPUT:
Name: Rani
Gender: Female
AttributeError: ‘Mydata'
object has no attribute
'__age'
```

Object-Oriented Approach in Python cont.

Key Points to Remember:

- Object-Oriented Programming makes the program easy to understand as well as efficient.
- Since the class is sharable, the code can be reused.
- Data is safe and secure with data abstraction.
- Polymorphism allows the same interface for different objects, so programmers can write efficient code.

Thank You