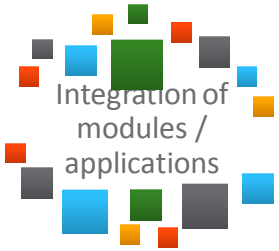
Day – 8
Object Oriented Programming with PYTHON

Session Plan

- Basic OOP concepts
- Creating classes and objects
- Class variables and Object Variables
- Method Invocation
- Using default arguments in Methods
- Static, Class and Instance Methods
- Relationships
    - Inheritance
    - Aggregation
    - Association

Need for Object Oriented Approach

- Challenges in developing a business application



Extensibility of existing code

High level of flexibility

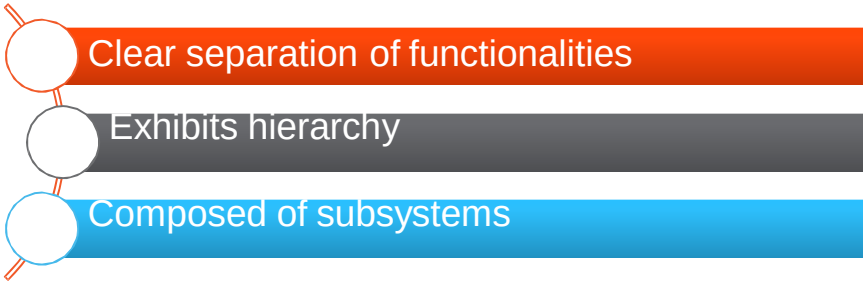
Illusion of simplicity

- If these challenges are not addressed it may lead to **Software Crisis**
- Features needed in the business application to meet these challenges:

- Challenges can be addressed using object oriented approach

Need for Object Oriented Approach

- Properties of a business application



- These properties can be implemented using object oriented approach

Easy Shop application is a complex business application & object oriented approach may be used to develop this system

OO Terminologies

Object	Anything that provides a way to locate, access, modify, and secure data
Class	A description of what data is accessible through a particular kind of object, and how that data may be accessed
Method	The means by which an object's data is accessed, modified, or processed
Abstraction	Focusing on "**what**" should be hidden
Encapsulation	Separate **what from the how.**
Inheritance	The way in which existing classes of object can be upgraded to provide additional data or methods
Polymorphism	The way that distinct objects can respond differently to the same message, depending on the class they belong to

Classes & Objects (1 of 2)

Classes & Objects (2 of 2)

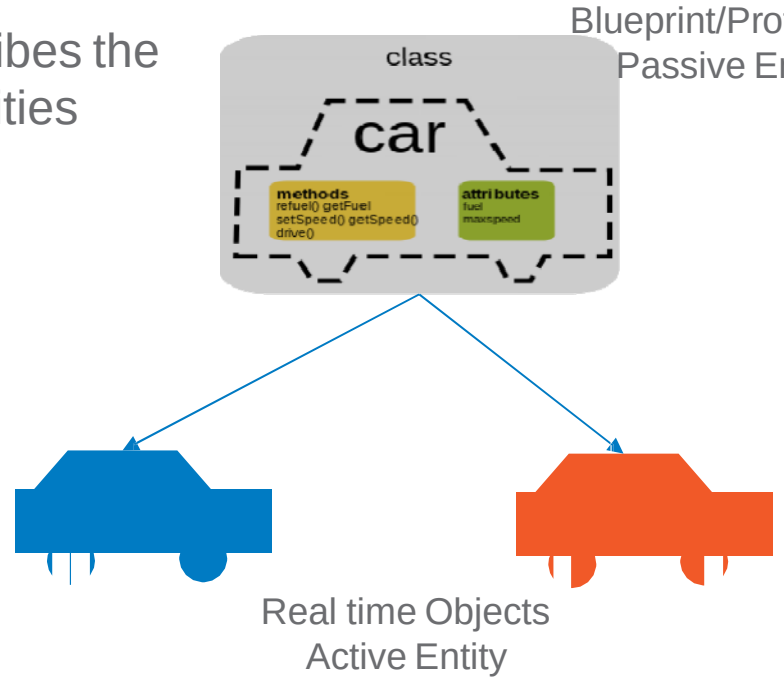
- A class is a prototype / design that describes the common attributes (properties) and activities (behaviors) of objects

Attributes

- Example : Customer Id, Name, Telephone number and Address

Behavior/ Activity

- Activities(behavior) exhibited by the class to external world
- Example: Purchasing items from the retail shop



Everything in Python is an object

- When you are working with Python, always remember that **everything i.e. variable, class, function, method etc. in Python is an object**
- Thus Python embraces OOP at a fundamental level
- An object consists of:
    1. A collection of related information i.e. attributes.
    2. A set of operations to manipulate that information i.e. behaviors / methods.

Abstraction – Guided Activity

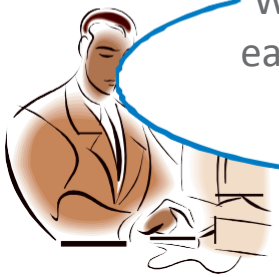
- Users of the retail application – Billing staff, Admin, Retail outlet manager
- Each user needs to know some details and need not know other details

Who are the users of the retail application?

What are the things each user must know to perform their activities ?

Billing staff (Billing of customers)

Admin (Registration of customers)

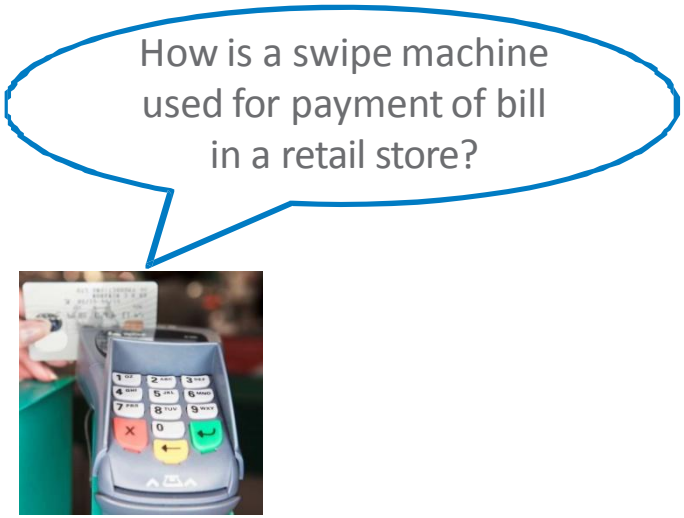


Retail Outlet Manager (Registration of users)

ABSTRACTION : Process of identifying the essential details to be known and ignoring the non-essential details from the perspective of the user of the system

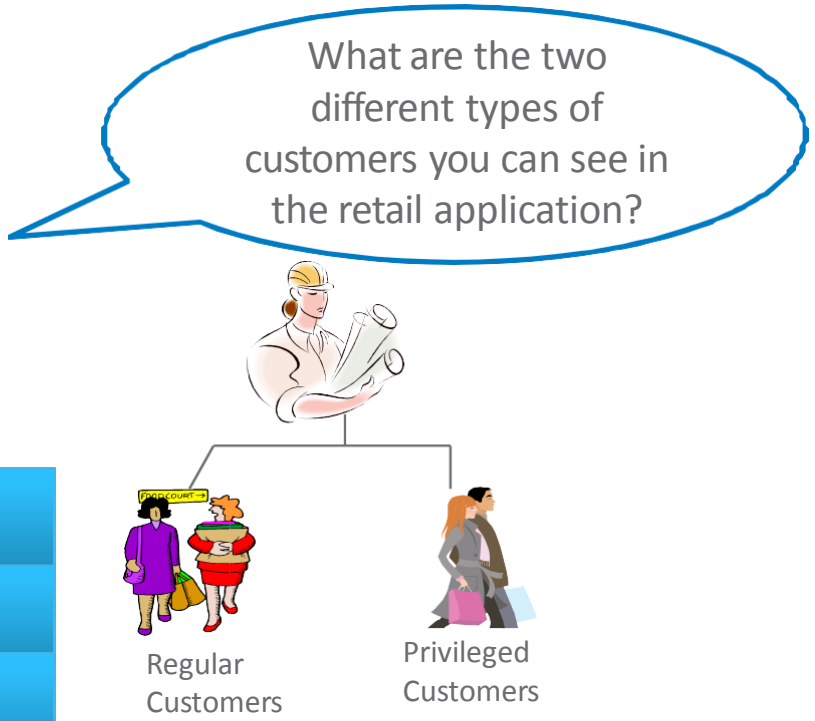
Encapsulation – Guided Activity

- Swipe machine in a retail store
  - Used by billing staff to key the amount
  - Used by admin to record payment



ENCAPSULATION : A mechanism of hiding the internal details and allowing a simple interface which ensures that the object can be used without having to know how it works

Inheritance – Guided Activity



- Customers are of two kinds
  - Regular
  - Privileged

All customers have Customer Id, Name, Telephone Number and Address

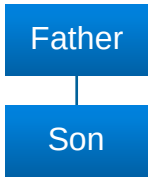
The regular customer in addition is given discounts

The privileged customer gets a membership card based on which gifts are given

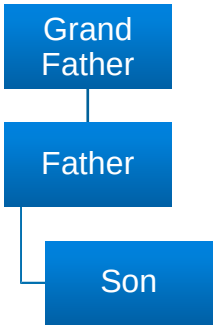
All customers have some generic features. The different kinds of customers have all generic features in addition to some specific features

INHERITANCE : Is a mechanism which allows to define generalized characteristics and behavior and also create specialized ones. The specialized ones automatically tend to inherit all the properties of the generic ones

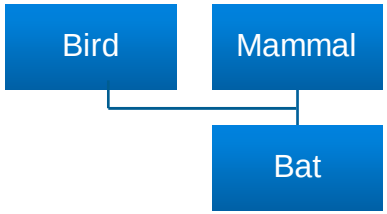
Types of Inheritance



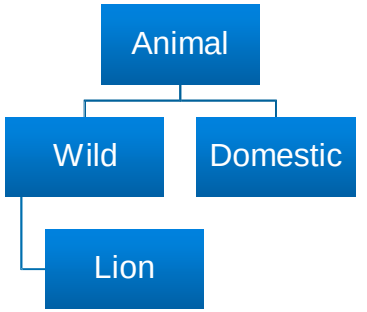
Single Inheritance



Multi-level Inheritance



Multiple Inheritance



Hierarchy Inheritance

Polymorphism – Guided Activity

- Payment of bill - Two modes
  - Cash (Calculation includes VAT)

What do you observe in this retail store scenario?

Total Amount = Purchase amount + VAT

  - Credit card(Calculation includes processing charge and VAT)

Total Amount = Purchase amount + VAT + Processing charge

POLYMORPHISM: Refers to the ability of an object/operation to behave differently in different situations

Object Oriented Approach – Benefits

- Leads to development of smaller but stable subsystems
- The subsystems are resilient to change
- Reduces the risk factor in building large systems as they are built incrementally from subsystems which are stable

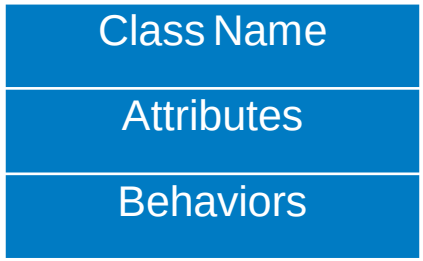
Hence Object Orientation is suitable for developing extremely complex business systems

Day – 9

Object Oriented Programming with PYTHON

Class Diagram

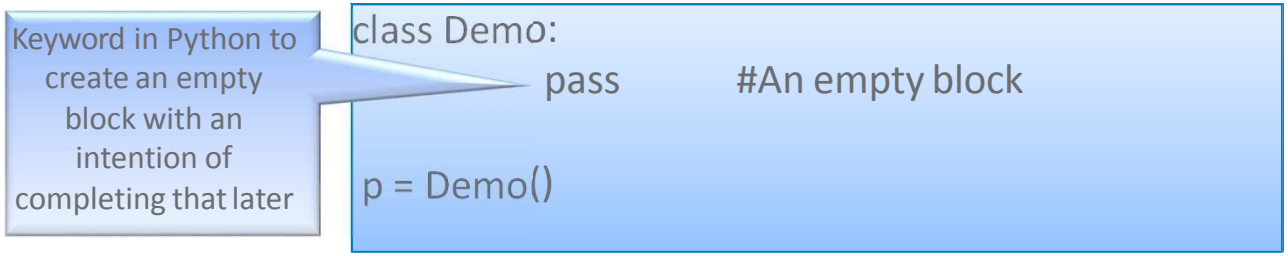
- Classes are the basic components of an object oriented system
- This diagram shows the collection of classes and the relationships among them
- In UML, any class is represented by a rectangular box divided with three compartments :



Access Specifiers
+ public
- private
protected

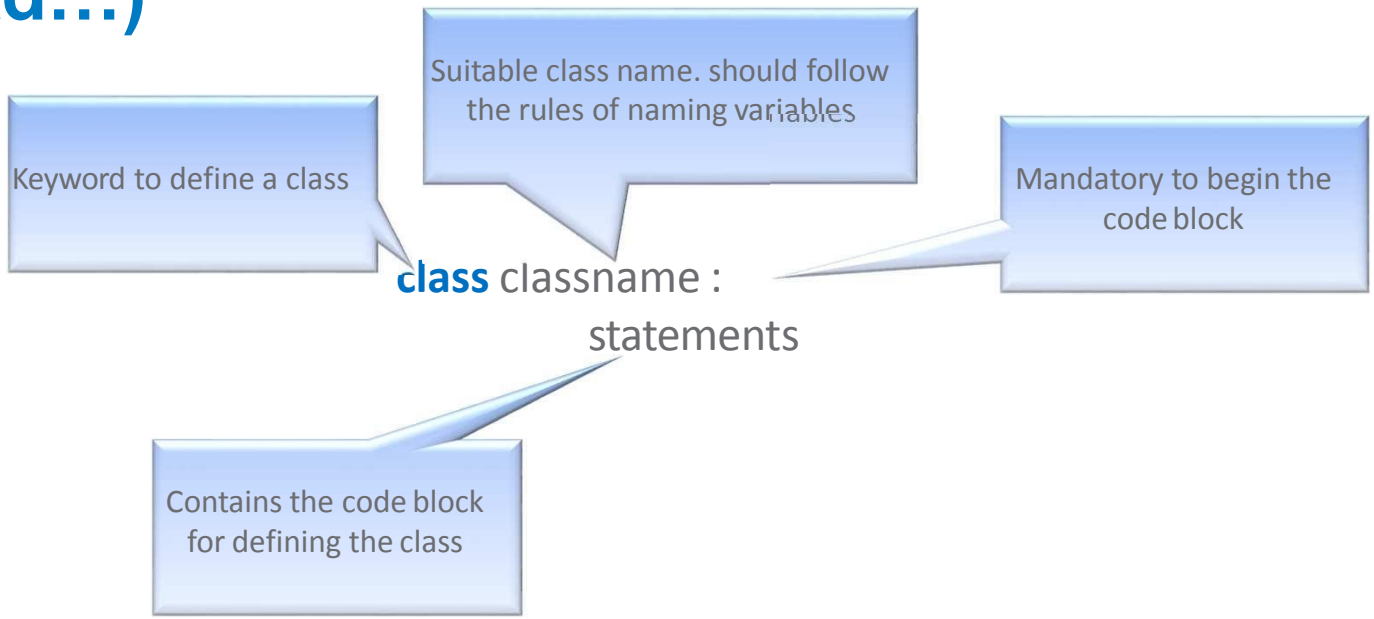
Creating Classes

- Classes are the main OOP tool in python
- It is created using **class** statement, contains **attributes** and **methods**
- An indented block of statements forms the body of the class.



- Like functions and modules, classes are also python program units but they are more useful while building new objects.

Creating Classes (contd…)



Access Specifiers

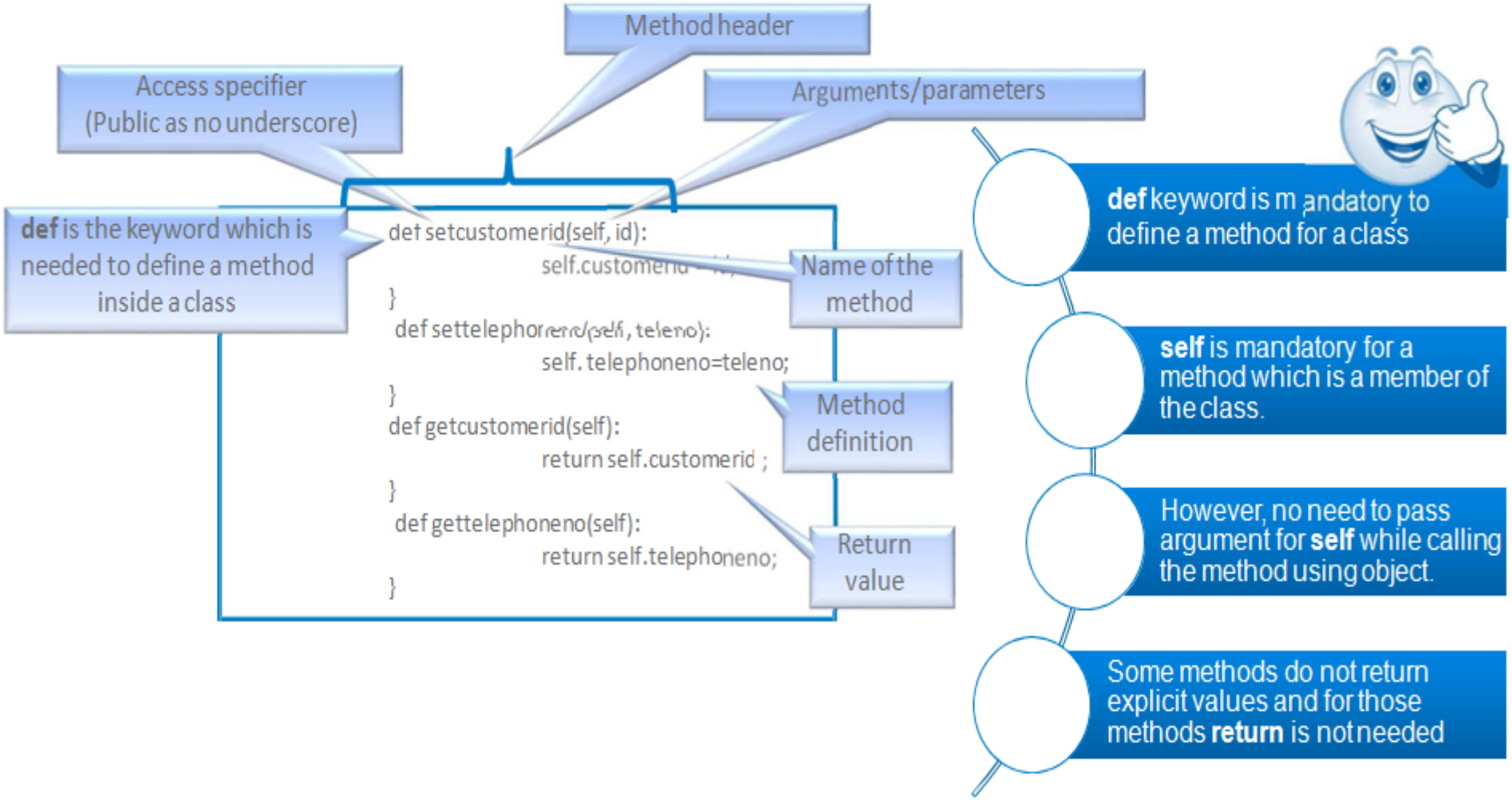
- Used to expose or hide the attribute and behavior of a class
- Used to specify the access permitted on a member

Naming	Type	UML Notation	Meaning
name	Public	+	These attributes can be freely used inside or outside of a class definition. A declaration that is accessible to all classes
_name	Protected	-	Protected attributes should not be used outside of the class definition, unless inside of a subclass definition.
__name	Private	#	This kind of attribute is inaccessible and invisible. It's neither possible to read nor write to those attributes, except inside of the class definition itself. A declaration that is accessible only to the class in which it is declared

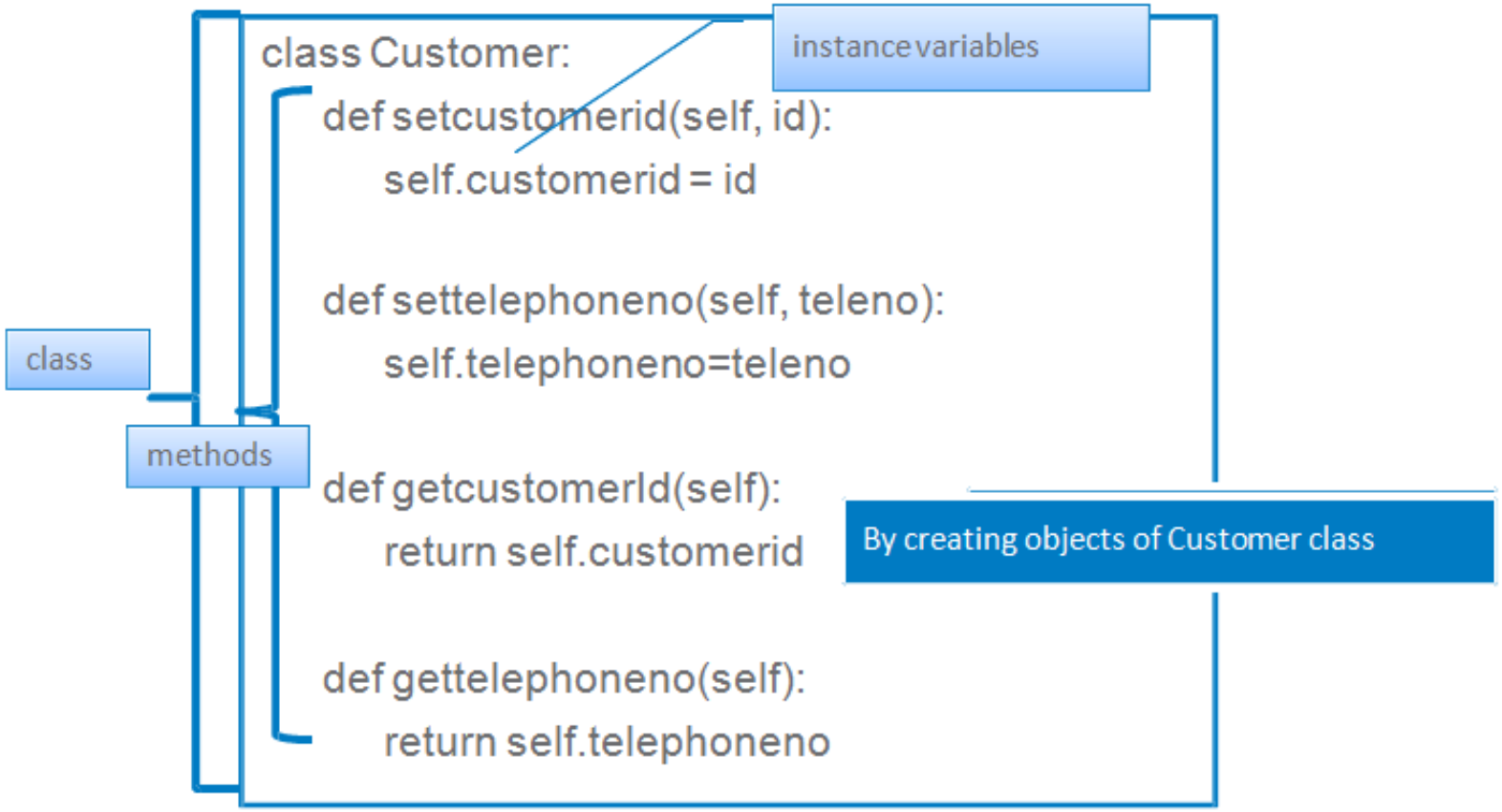
Methods (1 of 2)

- Methods define the behavior of an object
- All the methods should be defined inside the class with an access specifier
- Values that are passed to methods are known as arguments/parameters
- Value that is returned from a method is known as return value
- A method can return only one value at a time and it can be done using the return statement
- Implementation of a method requires the following:

Methods (2 of 2)



Implementation of a Class in Python – Guided Activity



Creation of Objects

- Class is a blueprint for the creation of objects
- To realize a class, an object or an instance of the class needs to be created
- There can be many instances for a class and each instance will have its own data
- In Python, Class name with brackets allocates memory for objects during run time i.e. dynamic memory allocation e.g.

```
Customer()
```

- Also one or more arguments could be passed for object creation e.g.

```
Customer(1001, "Kevin")
```

```
Customer(cid=1001, name="Kevin")
```

This will be discussed more in __init__() method.

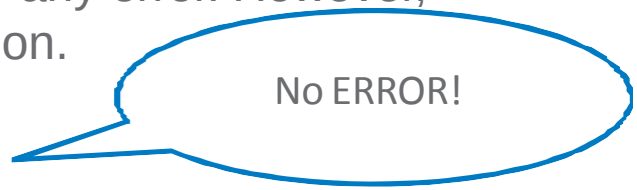
Reference Variables

- The reference returned by a newly created object must be tagged to a variable and that is known as reference variable
- Following syntax can be used to create a reference variable for the Customer class and make it point to a Customer object

```
custobj = Customer()
```

- As there is no need for declaration in Python, custobj will be identified as reference type automatically during the runtime.
- In python, if you end with semicolon (;) it will not display any error. However, ending semicolon is not in the statement syntax of python.

```
custobj = Customer();
```



Summary

- OOP in Python
  - Basic OOP concepts
  - Creating classes and objects
  - Class variables and Object Variables
  - Method Invocation
  - Using default arguments in Methods
  - Static, Class and Instance Methods
  - Relationships
    - Inheritance
    - Aggregation
    - Association