EdYoda Digital University

Python-21 March 2022

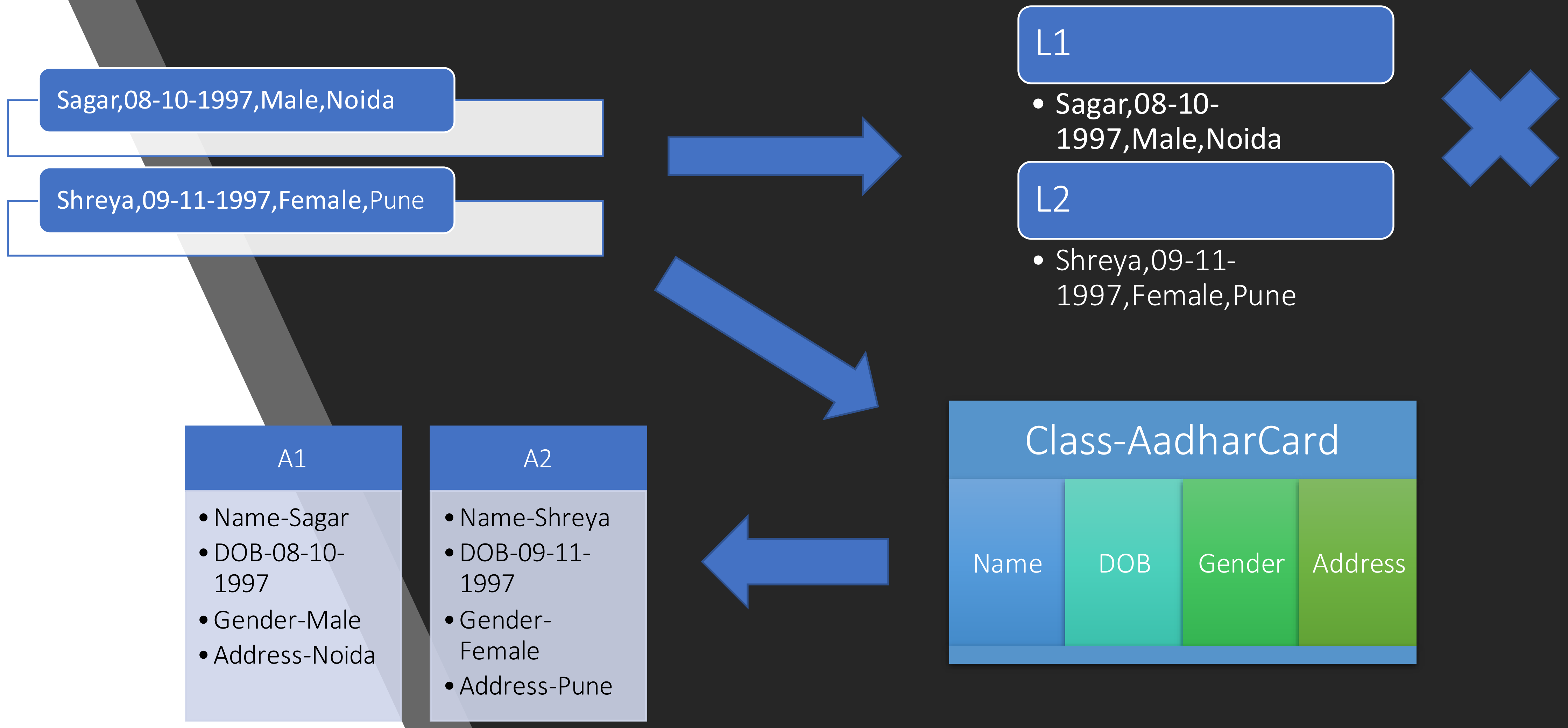
Batch-DS250322

Sagar Sarkar

Day 27- 9 May OOPs-1

- Introduction to Object Oriented Programming
- Classes and Objects
- Self
- __init__
- Class variable and Instance variables.

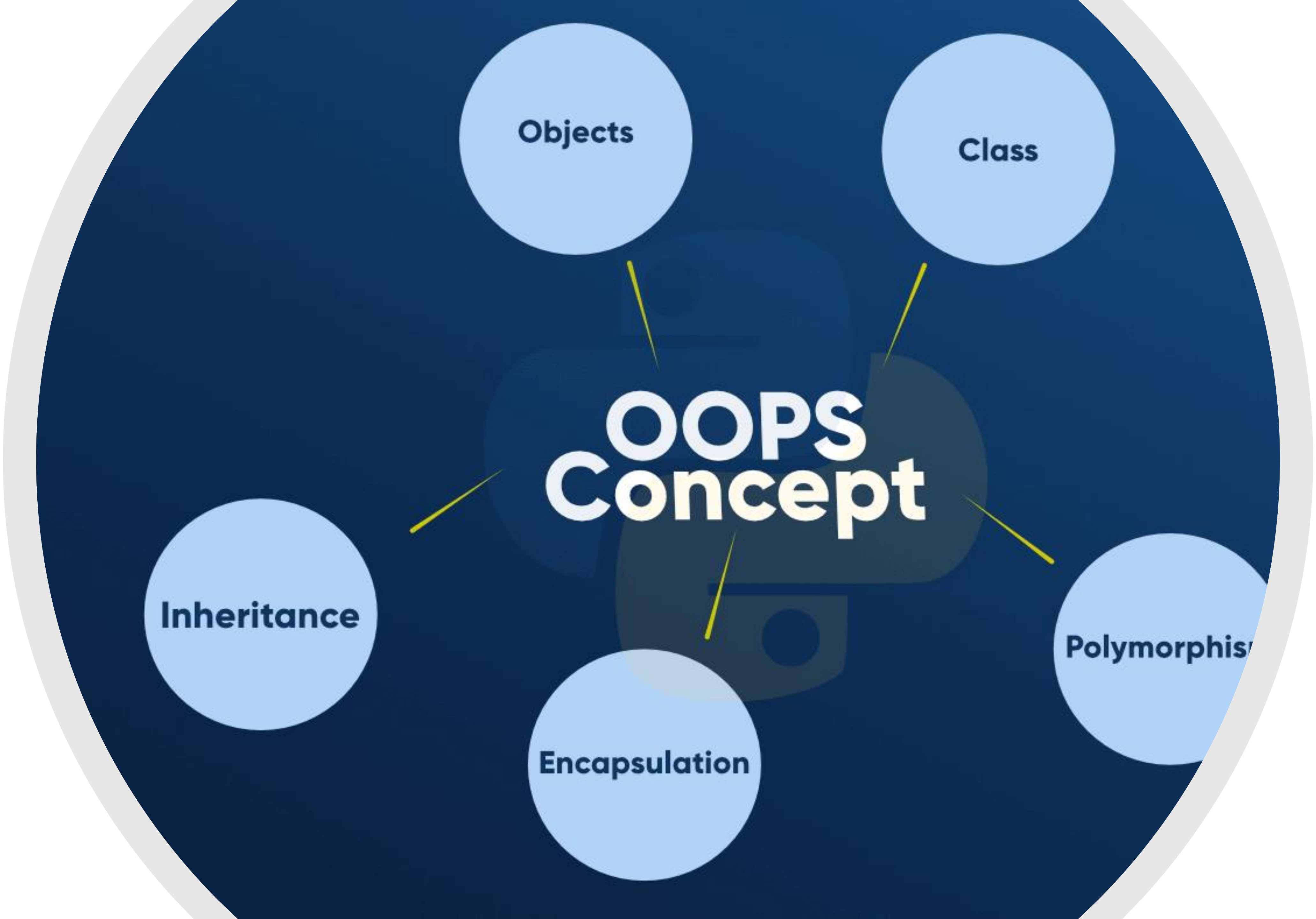
Why OOPs



Sagar,08-10-1997,Male,Noida
Shreya,09-11-1997,Female,Pune
L1
Sagar,08-10-1997,Male,Noida
L2
Shreya,09-11-1997,Female,Pune
Class-AadharCard
Name
DOB
Gender
Address
A1
Name-Sagar
DOB-08-10-1997
Gender-Male
Address-Noida
A2
Name-Shreya
DOB-09-11-1997
Gender-Female
Address-Pune

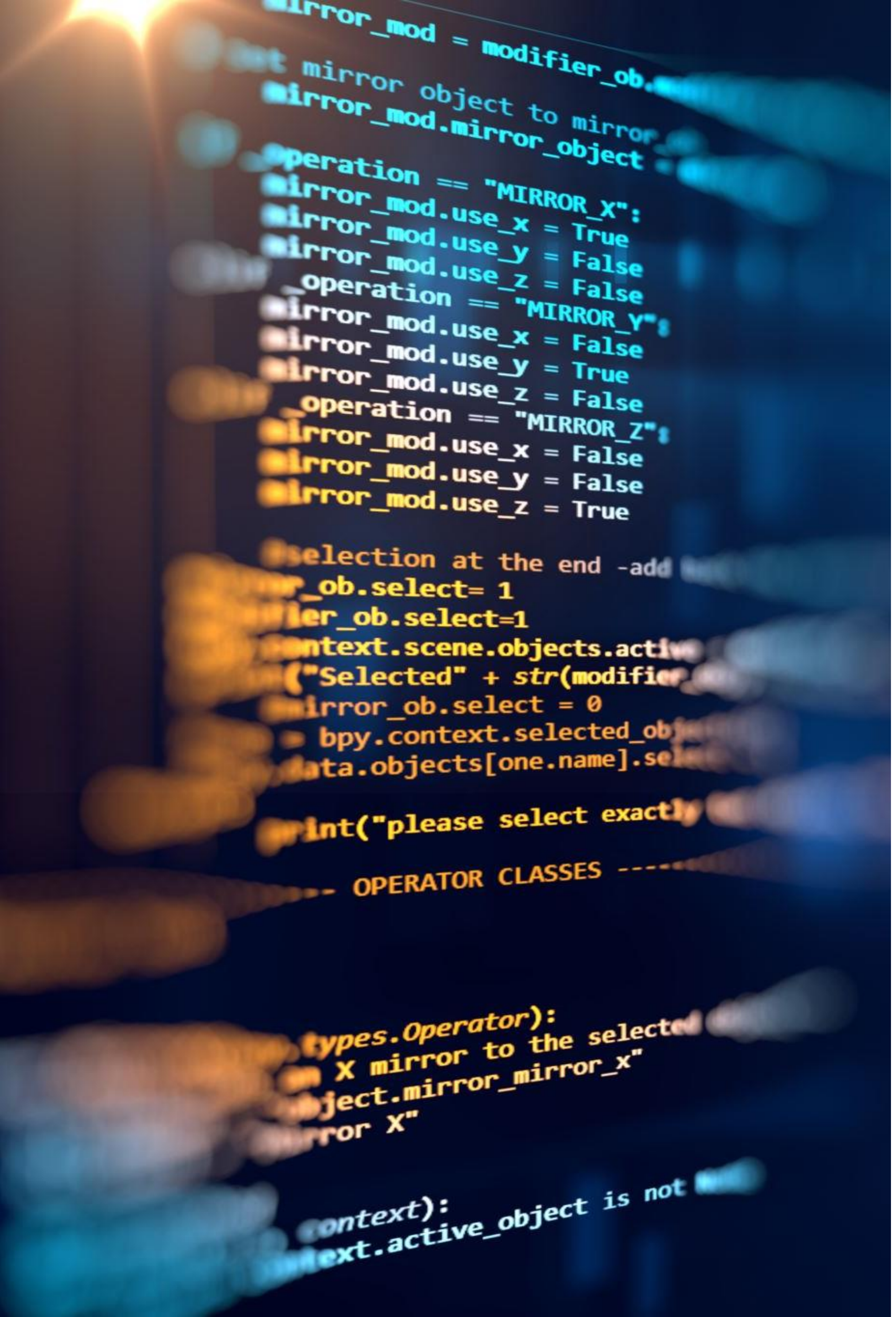
Introduction to Object Oriented Programming

- In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming.
- It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.
- The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.


Objects
Class
OOPS
Concept
Inheritance
Encapsulation
Polymorphis

Classes

- A class is a user-defined blueprint or prototype from which objects are created.
- Classes provide a means of bundling data and functionality together.
- Creating a new class creates a new type of object, allowing new instances of that type to be made.
- Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
- A class is like a blueprint for an object.



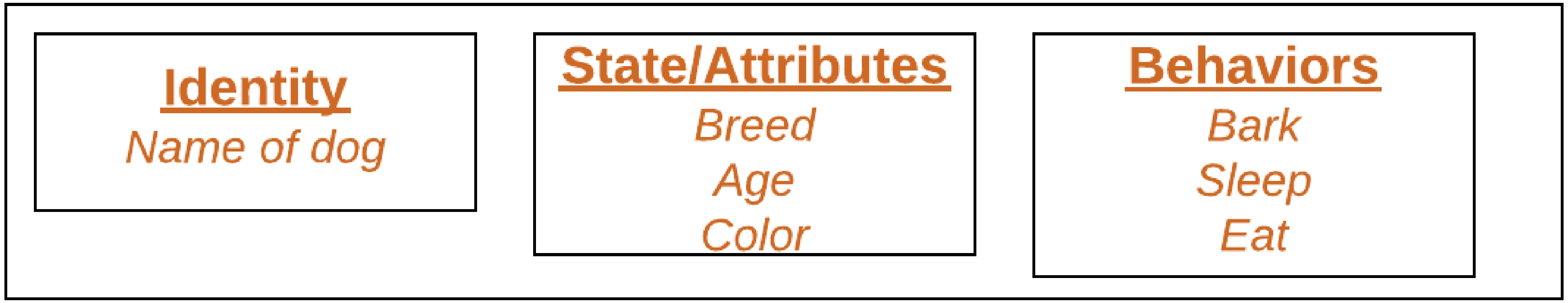
Class Definition Syntax:

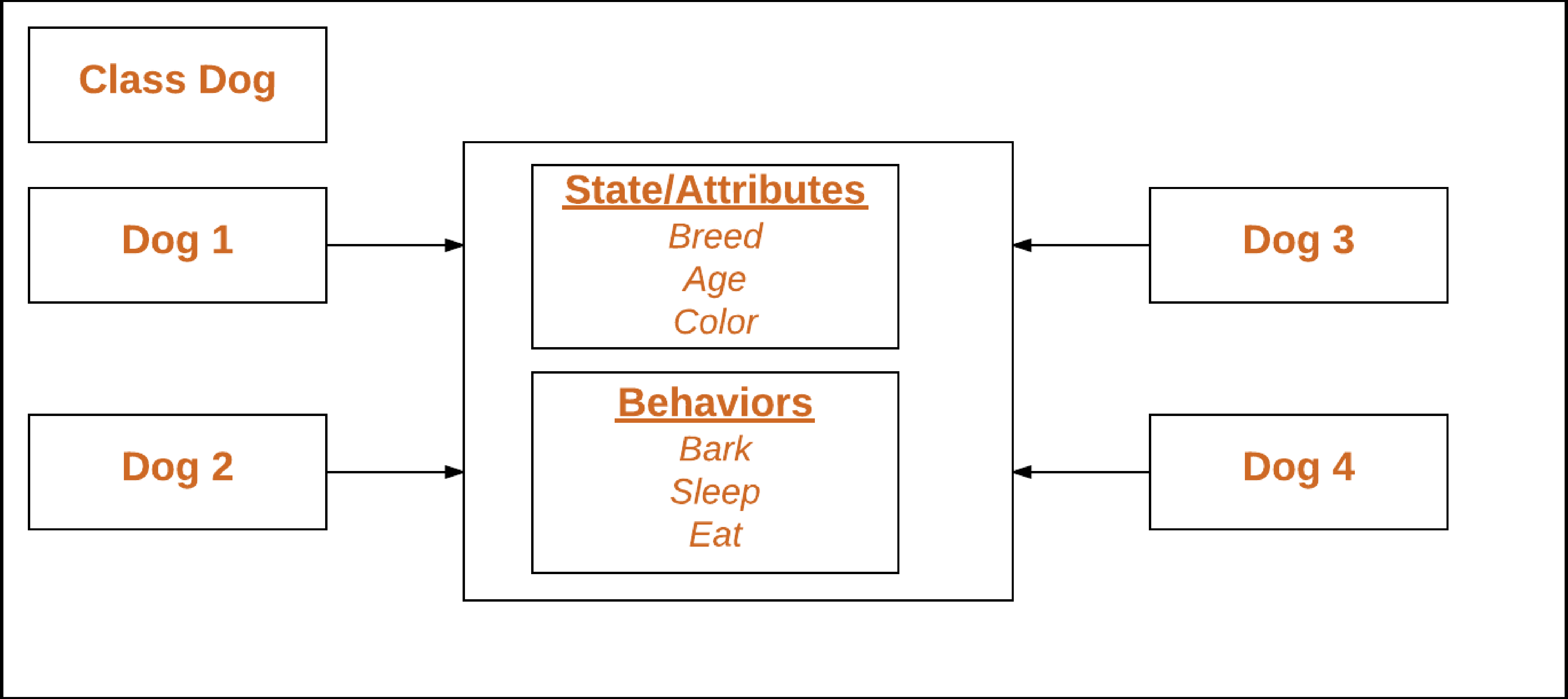
```
class ClassName:
    # Statement-1
    .
    .
    .
    # Statement-N
```

Objects

An object consists of :

1. **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
2. **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
3. **Identity:** It gives a unique name to an object and enables one object to interact with other objects.





Class Representation

Self

- Self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.
- Python decided to do methods in a way that makes the instance to which the method belongs be passed automatically, but not received automatically: the first parameter of methods is the instance the method is called on.
- *Self is always pointing to Current Object.*

Self

- Self must be provided as a First parameter to the Instance method and constructor. If you don’t provide it, it will cause an error.
- *Self is a convention and not a Python keyword .*
- But it is advisable to use self because it increases the readability of code, and it is also a good programming practice.

__init__

- The `__init__` method is similar to **constructors** in C++ and `Java`. Constructors are used to initialize the object’s state.
- The task of constructors is to initialize(assign values) to the data members of the class when an object of class is created.
- Like methods, a constructor also contains collection of statements(i.e. instructions) that are executed at time of Object creation.
- It is run as soon as an object of a class is instantiated.

__init__

- In the above example, a person name Nikhil is created. While creating a person, “Nikhil” is passed as an argument, this argument will be passed to the `__init__` method to initialize the object.
- The keyword `self` represents the instance of a class and binds the attributes with the given arguments. Similarly, many objects of Person class can be created by passing different names as arguments.

Polymorphism and Encapsulation

- Polymorphism

Polymorphism simply means having many forms.

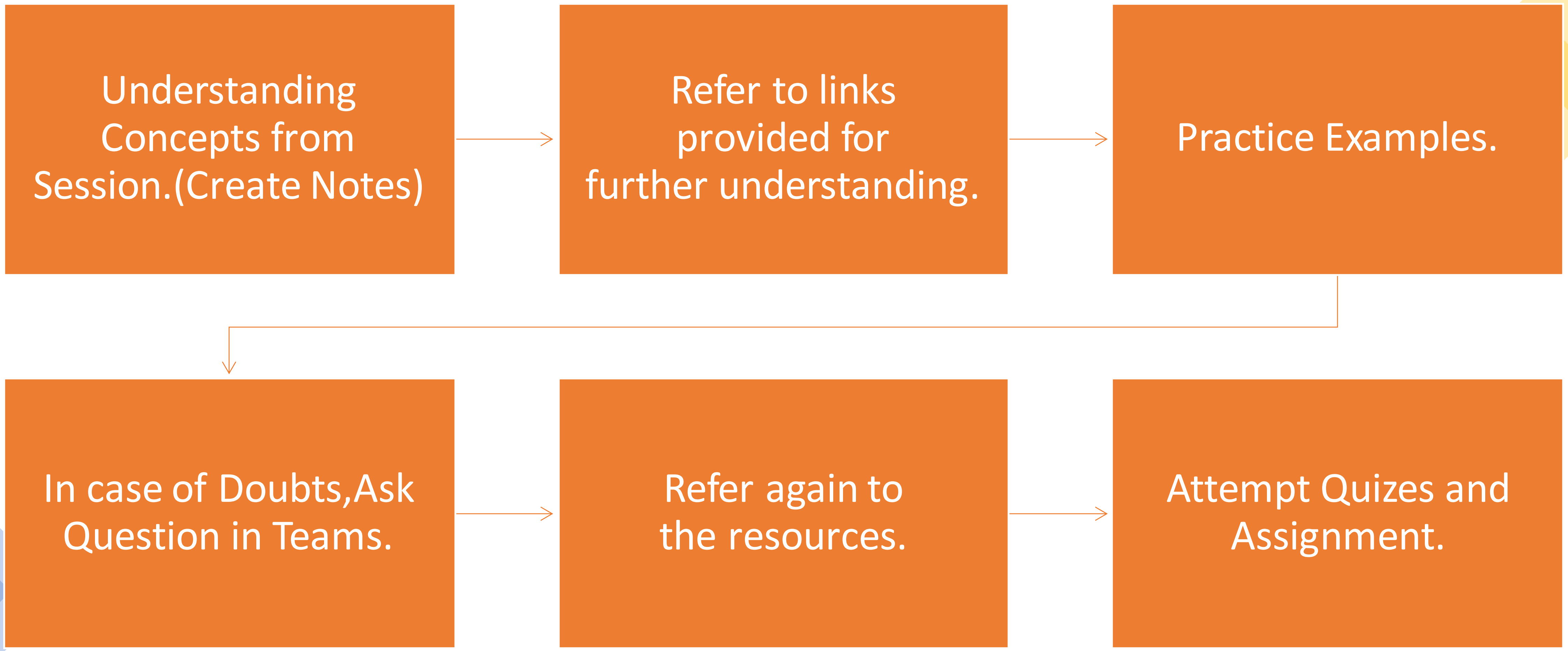
- Encapsulation

1. Encapsulation is one of the fundamental concepts in object-oriented programming (OOP).
2. It describes the idea of wrapping data and the methods that work on data within one unit.
3. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data.

Class variable and Instance variables

- Class attributes belong to the class itself they will be shared by all the instances.
- Such attributes are defined in the class body parts usually at the top, for legibility.
- Unlike class attributes, instance attributes are not shared by objects.
- Every object has its own copy of the instance attribute (In case of class attributes all object refer to single copy).

Approach to learning Python



Understanding Concepts from Session.(Create Notes)

Refer to links provided for further understanding.

Practice Examples.

In case of Doubts,Ask Question in Teams.

Refer again to the resources.

Attempt Quizes and Assignment.

Anyone ??