- Class is a blue print for creating new objects
- Every object (int, string etc) in python has a class which produces that objects
- An Object is an instance/example of a class
- Human is a class & Mosh, Brad etc are objects
- Naming convention for classes is CamelCase or Pascal
- All functions in our classes should have atleast one parameter called as self
- isInstance is a method to find the instance/object of a class
- Constructor is a special method which is called when we create a new object
- Magic method is known as constructor ➔ __init__
- Self is the reference to the current object
- Objects also have attributes which are basically variables containing data about a specific object
- Like javascript the objects in python are dynamic
- Class level attributes are shared across all instances/objects of that type
- Instance level attributes are limited to that instance or object only
- Factory method is called factory as it acts as a factory of manufacturing or creating object
- Design pattern ➔ Factory method helps in initializing the values which are complex and helps design clean code
- Cls is the default parameter when we define a class method
- @classmethod is called as decorator and it is used to enhance the behavior of an object or function
- Magic methods are called automatically by python interpreter
- __str__ magic method is used when we want to convert an object to string
- `__gt__` Defines behavior for the greater-than operator
- We do have Normal arithmetic operators like `__add__`
- Data structures are also known as container types
- We can design our own custom container types
- We use private members in a class to avoid accidental access outside
- Private members can be formed by prefixing __private
- The aim of private members is not security rather it's a warning
- We can still access a private member by using __dict__ magic method which holds all the attributes in this class
- Pythonic is refered to any best practice in which full potential of python is used unlike unpythonic
- A Property is an object that sits infront of an attribute and allows us get and set values of that attribute
- A property looks like a proper attribute from outside but internally has 2 methods which we call

- Getters
- Setters
- @Property decorator
- We want our classes and objects to be minimal with minimum number of functions or methods exposed to the outside
- DRY (Don't Repeat Yourself)
- Inheritance is the mechanism which allows us to define the common behaviors or common functions in one class and then inherit that in other classes
- Parent-Base | Child – Sub
- Object class is the base class and every class we declare in python directly or indirectly is derived from the Object class
- IsSubClass method is used to verify an object is a sub class of other
- Method overriding is replacing method of base class in child class
- We can avoid method overriding by the use of super() function to access and change base class
- Inheritance is a good thing it helps avoid code repetition and allows code reusability
- However, too much of a good thing is a bad thing→ Too much inheritance between classes causes complexity in software
- Inheritance abuse → multi level inheritance ➔ Avoid it → Just upto 2-3 levels
- If you don't handle the multiple inheritance properly you will be going to invite all sort of bugs
- Flyer + Swimmer ➔ FlyerFish Example of Multiple Inheritance
- We can make custom error exception we can derive it from base exception class in python
- A good example of inheritance in real world is handling data from a data stream, may be from file on hard drive, a process on memory or a network stream
- An abstract base is like a half baked cookie, it provides some common code to it's derivatives
- Abc is AbstractBaseClass
- Abstract and Concrete classes
- Abstract base classes don't allow instances
- Concrete base classes allow instances to be developed
- Polymorphism ➔ Poly → Many | Morps → Forms
- Polymorphism is usage of a method in several forms
- Example: Draw() is being used in different forms as draw(Interface), draw(toolbar) etc
- Duck Typing➔ If it walks like a duck & quacks like a duck! It is a duck

- Duck typing is supported by python because it's dynamically typed language & it doesn't check the type of object, it only looks for the existence of certain method is available or not
- We can easily extend the functionality of built-in data types such as lists & strings etc.
- We use classes to build data and functionality in one Unit
- Id() function provides the address of memory location where an object is stored
- Those classes which do not have any behavior/functionality are referred to as "Data Classes"
- If you are working with Data Classes it's better practice to use NamedTuples instead of classes