class

- collection of attributes and methods (functions)
  - attributes
    - characteristic of an entity
  - methods
    - function inside a class
- blueprint to create an object

object

- instance of a class
- memory allocated to store the class attributes
- charactertistics
  - state: values of attributes
  - behavior: defined by methods in the class
  - identity: unique address

instantiation

```
* process of creating an object
```

attributes

- In Python, each object may have different attributes (even if objects are of the same class).
  - To attach an attribute to an object, use setattr().
  - To retrieve attribute value, use getattr().
  - To remove the attribute from object, use delattr().
  - To see if object has a attribute, use hasattr().

class methods

- function declared inside the class
- each method gets an implicit arg "self". It keeps address of the object on which the method is called.
- method types
  - initializer
    - method which initializes the object
    - gets called automatically/implicitly
  - facilitator
    - adds facility in the class
  - setter
    - used to set a new value to an attribute
    - also known as mutator
  - getter
    - used to get a value of an attribute

        - also known as inspector
    - de-initalizer (delete)
        - used to de-initialize the object
        - gets called automatically/implicitly
    - dunder methods
        - special methods to be written in class for special purposes.

inheritance

- also known as is-a or kind-of relationship
- the base class can be specified at the time of declaration with ( )
- Inheritance types
    - single inheritance
        - there is only one base class and only one derived class
    - multi-level
        - there are multiple levels
        - a level will have one base and one derived class
        - a may have one direct and mulitple indirect base class(es)
    - multiple
        - multiple base classes
        - one derived class
    - hierarchical
        - one base class and multiple derived classes
    - hybrid
        - combination of any two or more inheritance types is known as hybrid.

method overriding

- in inheritance scenario, when derived class uses method with same name as that of the base class
- used to change the behavior/implementation of base class method
- to overide a method in derived class, use same name of the method as that of the base class

root class

- in python (3.0+), object is a root class
- in python, every class is a derived class of object (directly or indirectly)
- object class provides basic functionality like
    - converting any object to string (**str**)

Abstract base class (ABC)

- Sometimes developer of a class wants to ensure that all classes derived from the class must have certain functionality (like a contract).
- Such methods are declared as abstract methods (in base class). In Python, this is done using @abstractmethod decorator (from abc package). Usually, these methods are empty (unimplemented) in the base class.
- If a class contains at least one @abstractmethod, then the class must be inherited from ABC class (from abc package). The object of such abstract class cannot be created (it raise TypeError).

- The classes inherited from such abstract class must provide implementation of the abstract methods; otherwise derived class objects cannot be created.
- Note that abstract classes may have attributes and other methods.

Association

- represents associations of multiple classes
- types
  - aggregation
    - also known as has-a relationship
    - loose coupling / weak relationship
    - one entity can live without another entity
    - one object contains an object of another class
  - composition
    - also known as composed-of / part-of
    - tight coupling / strong relationship
    - one entity can not live without the other entity

Operator Overloading

- changing the default behavior of built-in operators
- the methods to be implemented in class are called as magic methods.
- comparison operators
  - p1 > p2 : **gt**
  - p1 < p2 : **lt**
  - p1 <= p2: **le**
  - p1 == p2: **eq**
  - p1 != p2: **ne**
  - p1 >= p2 : **ge**
- mathematical operators
  - p1 + p2 : **add**
  - p1 – p2 : **sub**
  - p1 * p2 : **mul**
  - p1 / p2 : **truediv**
  - p1 // p2: **floordiv**
  - p1 ** p2 : **pow**
  - p1 % p2 : **mod**