1. modular programming

reusability

inheritance

2. Attribute fetches are simply tree searches. The term inheritance is applied because objects lower in a tree inherit attributes attached to objects higher in that tree. As the search proceeds from the bottom up, in a sense, the objects linked into a tree are the union of all the attributes defined in all their tree parents, all the way up the tree.

3. Objects and Instances are almost similar and are often used interchangeably but with a small difference. Object is a generic term , it is physically present but remains undiferrentiated. Instance is something that gives them a separate identity.

Object is the physical entity for which memory is allocated. Object contains many instances.

Instance : An instance is also the physical manifestation of a class that occupies memory and has data members.

e.g. There is Class car, when I create c = car(), c is object. When we create different object with different specifications(car name, type, company) such as i10, i20, Creta, audi7 are called as instances which actually exists.

4. Self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python

5. The task of init method is to initialize(assign values) to the data members of the class when an object of class is created.

6. To create instances of a class, you call the class using class name and pass in whatever arguments its init method accepts.

7. class Student:

def \_\_init\_\_(self,name):

self.name = name

def \_\_str\_\_(self):

return f"This is {self.name} (an instance of a class Student)"

student1 = Student("Mahak")

print(student1)

8. A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class.

class A:

def method\_A(self):

return "this is method of class A"

class B(A):

def method\_B(self):

return "this is method of class B"

class C(A):

def method\_B(self):

return "this is method of class C"

class D(A):

def method\_B(self):

return "this is method of class D"

# Class B, C, D are subclasses and Class A is superclass

a = A()

b = B()

c = C()

d = D()