1. Multiple Inheritance

When a class is derived from more than one base class it is called multiple Inheritance. The derived class inherits all the features of the base case.

2. Delegation is an object oriented technique (also called a design pattern). Let's say you have an object x and want to change the behaviour of just one of its methods. You can create a new class that provides a new implementation of the method you're interested in changing and delegates all other methods to the corresponding method of x.

Python programmers can easily implement delegation. For example, the following class implements a class that behaves like a file but converts all written data to uppercase:

class UpperOut:

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

self.\_outfile = outfile

def write(self, s):

self.\_outfile.write(s.upper())

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

return getattr(self.\_outfile, name)

Here the UpperOut class redefines the write() method to convert the argument string to uppercase before calling the underlying self.\_\_outfile.write() method. All other methods are delegated to the underlying self.\_\_outfile object. The delegation is accomplished via the \_\_getattr\_\_ method; consult the language reference for more information about controlling attribute access.

3. Composition is one of the important concepts of Object-oriented programming (OOPs). Composition basically enables us for creating complex types objects by combining other types of objects in the program. Composition represents ‘has a relationship’ type or we can call composition as ‘has a relationship’ in the OOPs concept. It means that a composite class present in the program can contains the objects from the other class components and this type of relationship model is called as has a relationship.

Note: The classes that contains objects from the other classes in the program are called as composites whereas the classes which we use to create more complex type of relationship models is called as components.

4. If a function is an attribute of class and it is accessed via the instances, they are called bound methods. A bound method is one that has ‘ self ‘ as its first argument. Since these are dependent on the instance of classes, these are also known as instance methods.

5. Python today does support the notion of name "mangling" (i.e., expansion) to localize some names in classes. Mangled names are sometimes misleadingly called "private attributes," but really this is just a way to localize a name to the class that created it—name mangling does not prevent access by code outside the class. This feature is mostly intended to avoid namespace collisions in instances, not to restrict access to names in general; mangled names are therefore better called "pseudoprivate" than "private."

Pseudoprivate names are an advanced and entirely optional feature, and you probably won't find them very useful until you start writing general tools or larger class hierarchies for use in multiprogrammer projects. In fact, they are not always used even when they probably should be—more commonly, Python programmers code internal names with a single underscore (e.g., \_X), which is just an informal convention to let you know that a name shouldn't be changed (it means nothing to Python itself).