Object

Every real-world entity is an object.An object has Behaviour (things it does or performs) and Attributes (things that describe it).
For eg: A Chair object can have behaviour like Movement, Height Adjustment & Attributes like Color, Make & Model, and Price.

Encapsulation

It means wrapping data into a single unit & securing it.
For eg: Drug Capsule wraps different medicines into a single unit and protects them from the outside environment.
Bank Locker wraps your valuables into a single unit(locker) and protects it via passcode.

Abstract class/Method

Abstract class is a class that cannot be instantiated. However, you can create classes that inherit from an abstract class. An abstract method is an method without an implementation. An abstract class may or may not include abstract methods.

Python doesn't directly support abstract classes. But it does offer a module that allows you to define abstract classes.To define an abstract class, you use the abc (abstract base class) module.

Ex:
```
from abc import ABC, abstractmethod
class Polygon(ABC):
@abstractmethod
def noofsides(self):
pass
```

Class

The collection of all related objects is called a class.Consider class as a general category which contains all the related objects inside it.
For eg: Objects like Wheelchair, Office Chair and Wooden Chair can be a part of the "Chair" class.

Abstraction

Hiding complexity from the user and showing only the relative stuff.
For Eg: In Car, all the complexity like the engine, machinery, etc is hidden from you; only relevant parts are shown, like the brakes, accelerator, and gearbox.

Generators

Generators are functions that return an iterable generator object. Because the values from the generator object are fetched one at a time rather than the entire list at once, you can use a for-loop, next(), or list() function to get the actual values. Generator functions act just like regular functions with just one difference that they use the Python yield keyword instead of return .
Code:
```
def test_sequence():
num = 0
while num<10:
yield num
num += 1
for i in test_sequence():
print(i, end=",")
```
Output: 0,1,2,3,4,5,6,7,8,9

Inheritance

The way we inherited a few qualities from our parents similarly, a class can also inherit the qualities from a parent class.

For eg: A Phone Class can have two Child Classes: 1) TelePhone
and 2) MobilePhone. Both can inherit the "-calling" behaviour.

Different types of Inheritance:
Single inheritance: When a child class inherits from only one parent class, it is called single inheritance. We saw an example above.
Multiple inheritances: When a child class inherits from multiple parent classes, it is called multiple inheritances.
Multilevel inheritance: When we have a child and grandchild relationship.
Hierarchical inheritance: More than one derived class are created from a single base.
Hybrid inheritance: This form combines more than one form of inheritance. Basically, it is a blend of more than one type of inheritance.

Polymorphism

It means many forms. With the same name, it provides different forms.
For eg: In Chess, we've 6 pieces - king, rook, bishop, queen, knight, and pawn. All of them "move" differently i.e. Bishop moves diagonally, Rooks move horizontally and vertically, etc.