1. **Operator overloading &Method overriding.**
2. **Static attribute, Static method & Class method decorator:**
   @staticmethod
   Normal Method outside of the class.can access via direct class without any instance.
   @classmethod
   will ignore “self” parameter.

3. **Getter, Setter and Property, Read only : Using Property Decorator**

   getter without any setter is read-only attribute.
   @property : conver a method in a attribute.
   [By default its a getter][Read-only]

   To use setter we must have getter.
   @GetterName.setter

   i. read only --> you can not set the value. value can not be changed
   ii. getter --> get a value of a property through a method. Most of the time, you will get the value of a private attribute.
   iii. setter --> set a value of a property through a method. Most of the time, you will set the value of a private property.

4. **Inner function and wrapper function.**
   Function can be called in a inside function also, function can be send as a parameter in another function.

5. **How does decorator works :** Decorator call the other nested functions without passing any parameter and multiple calls()().

6. **Class Composition : inheritance vs composition.**
   Composition :
   class Engin:
   def __init__(self):
   pass

```
# car "has a" engine
class Car:
        def __init__(self) -> None:
                self.engine = Engine()
        def start(self):
                self.engine.start()
```

7. **UML Diagrams : Unified Modeling Language**
   1. Class Diagram
   2. Use Case Diagram
   3. Sequence Diagram
   4. Activity Diagram

8. **Design Patterns: Singleton, Factory, Builder, etc**
   1. Creational Patterns.
   2. Structural Patterns.
   3. Behavioral Patterns.
   4. Architectural pattern.