Data Abstraction

1. It hides unnecessary code details from the user.
2. Also, when we do not want to give out sensitive parts of our code implementation and this is where data abstraction came.
3. Abstraction is used to hide the internal functionality of the function from the users.
4. The users only interact with the basic implementation of the function, but inner working is hidden.
5. User is familiar with that **"what function does"** but they don't know **"how it does'.**
6. A method becomes abstract when decorated with the keyword @abstractmethod.
7. An Abstract class can contain the both method normal and abstract method.
8. An Abstract cannot be instantiated; we cannot create objects for the abstract class.

```
# Python program showing abstract base class work
from abc import ABC, abstractmethod
class Polygon(ABC):
    @abstractmethod
    def noofsides(self):
        pass
class Triangle(Polygon):
    # overriding abstract method
    def noofsides(self):
        print("I have 3 sides")
class Pentagon(Polygon):
    # overriding abstract method
    def noofsides(self):
        print("I have 5 sides")
class Hexagon(Polygon):
    # overriding abstract method
    def noofsides(self):
        print("I have 6 sides")
class Quadrilateral(Polygon):
    # overriding abstract method
    def noofsides(self):
        print("I have 4 sides")
# Driver code
R = Triangle()
R.noofsides()
K = Quadrilateral()
K.noofsides()
R = Pentagon()
R.noofsides()
K = Hexagon()
K.noofsides()
```

13. Difference between method **overloading** and **overriding**?

Method overloading	Method overriding
1. More than one method with same name, different prototype in same scope is called method overloading.	1. More than one method with same name, same prototype in different scope is called method overriding.
2. In case of method overloading, parameter must be different.	2. In case of method overriding, parameter must be same.
3. Method overloading is the example of compile time polymorphism.	3. Method overriding is the example of run time polymorphism.
4. Method overloading is performed within class.	4. Method overriding occurs in two classes.
5. In case of method overloading, Return type can be same or different.	5. In case of method overriding Return type must be same.
6. Static methods can be overloaded which means a class can have more than one static method of same name.	6. Static methods cannot be overridden, even if you declare a same static method in child class it has nothing to do with the same method of parent class.
7. Static binding is being used for overloaded methods	7. dynamic binding is being used for overridden/overriding methods.

Programming Exercise

1. Write a Python program to create a class representing a Circle. Include methods to calculate its area and perimeter.
2. Write a Python program to create a person class. Include attributes like name, country and date of birth. Implement a method to determine the person's age.
3. Write a Python program to create a calculator class. Include methods for basic arithmetic operations.
4. Write a Python program to create a class that represents a shape. Include methods to calculate its area and perimeter. Implement subclasses for different shapes like circle, triangle, and square.
5. Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.
6. Write a Python program to create a class representing a bank. Include methods for managing customer accounts and transactions.

OOP related programming exercise

https://www.w3resource.com/python-exercises/oop/index.php