**Лабораторная работа N7 ООП**

1. Что такое класс?
2. Что такое объект?
3. Как связаны между собой классы и объекты в программе?
4. Что собой представляет конструктор \_\_init\_\_()?
5. Зачем нужен конструктор?
6. Что такое инкапсуляция?
7. Как инкапсуляция реализована в Python?

**1.Класс (class) в программировании - это шаблон или описание для создания объектов. Он определяет атрибуты (переменные-члены) и методы (функции-члены), которые будут унаследованы объектами, созданными на основе этого класса. Классы являются основой объектно-ориентированного программирования (ООП) и позволяют организовать данные и функциональность в одной сущности.**

**2.Объект (object) - это экземпляр класса, созданный на основе определенного класса. Объекты представляют конкретные данные и состояние, а также могут выполнять методы, определенные в классе. Каждый объект класса имеет свои собственные значения атрибутов, но разделяет методы класса.**

**3.Классы и объекты связаны так, что класс определяет структуру и поведение объектов. Класс служит в качестве модели или чертежа для создания объектов. Объекты, созданные на основе класса, наследуют его атрибуты и методы. Это означает, что каждый объект может иметь свои собственные значения для атрибутов, но использует методы, определенные в классе.**

**4.Конструктор \_\_init\_\_() - это метод класса, который выполняется при создании нового объекта на основе класса. Этот метод инициализирует атрибуты объекта, задавая им начальные значения. Он обычно принимает self в качестве первого параметра, что обозначает создаваемый объект, и далее может принимать другие параметры, необходимые для инициализации.**

**5.Конструктор \_\_init\_\_() нужен для того, чтобы установить начальные значения атрибутов объекта. Он позволяет задать состояние объекта при его создании. Без конструктора объекты могли бы быть созданы, но они были бы пустыми или имели бы значения по умолчанию.**

**6.Инкапсуляция - это один из принципов ООП, который означает скрытие деталей реализации объекта и предоставление доступа к ним только через установленные интерфейсы. Это помогает упростить управление объектами и избежать прямого доступа к их внутренним данным. Инкапсуляция также позволяет сокрыть сложность реализации и предоставить более абстрактное представление объектов.**

**7.В Python инкапсуляция реализована с использованием концепции "закрытых" и "защищенных" атрибутов и методов. Все атрибуты и методы класса доступны извне по умолчанию, но можно задать их доступность с помощью префиксов:**

**Атрибуты и методы, начинающиеся с одного нижнего подчеркивания (например, \_private\_var), считаются "защищенными". Они не должны использоваться извне класса, но доступ к ним все равно возможен.**

**Атрибуты и методы, начинающиеся с двух нижних подчеркиваний (например, \_\_private\_var), считаются "закрытыми". Они более строго инкапсулированы и доступ к ним ersd возможен только внутри класса.**

**Однако в Python инкапсуляция основана на соглашениях и не обладает строгой защитой, так как все атрибуты и методы всегда могут быть доступны, если есть необходимость.**