**Контрольные вопросы:**

1. Проблемы императивного (процедурного) подхода в программировании.  
   ответ: Проблемы императивного (процедурного) подхода в программировании включают нарушение инкапсуляции данных, сложность поддержания и расширения кода, а также возможность возникновения конфликтов имен в глобальной области видимости.
2. Понятие объекта и черного ящика (примеры). Парадигма объектно-ориентированного программирования. Цель объектно-ориентированного подхода. Как выглядит разработка приложения в объектно-ориентированном стиле?  
   ответ: Понятие объекта и черного ящика: объект - это экземпляр класса, который содержит данные и методы для их обработки. Черный ящик представляет собой концепцию объектно-ориентированного программирования, когда внутренняя реализация объекта скрыта от пользователя, и он может взаимодействовать с объектом только через определенный интерфейс. Цель объектно-ориентированного подхода - создание модульного, гибкого и легко поддерживаемого программного обеспечения.
3. Понятия класс, объект, поле, метод. Область видимости, виртуальность метода.

Ответ: Понятия класс, объект, поле, метод: класс - это шаблон для создания объектов, объект - экземпляр класса, поле - переменная, хранящая данные объекта, метод - функция, принадлежащая определенному классу. Область видимости определяет доступность атрибутов и методов в коде. Виртуальность метода позволяет переопределить метод в подклассе.

1. Основные принципы ООП. Содержание и примеры.

Ответ: Основные принципы ООП включают в себя наследование, инкапсуляцию и полиморфизм. Наследование позволяет создавать новые классы на основе существующих, перенимая их свойства и методы. Инкапсуляция скрывает детали реализации от пользователя, обеспечивая безопасный доступ к данным. Полиморфизм позволяет использовать один и тот же интерфейс для различных типов данных.

1. Поддержка ООП в Python. Определение простого класса (инициализация, строковое представление, специальные методы).

Ответ: Поддержка ООП в Python включает в себя определение классов, их инициализацию, использование специальных методов (init, str), а также поддержку наследования и полиморфизма.

1. Определение операторов в классах, проверка типов. Случаи, в которых добавление такой функциональности является целесообразным.

Ответ: Определение операторов в классах и проверка типов может быть целесообразным в тех случаях, когда требуется определить пользовательские семантики для операций или обеспечить типовую безопасность в приложении.

1. Атрибуты объекта и атрибуты класса: ключевые различия и варианты использования.

Ответ: Атрибуты объекта относятся к конкретному экземпляру класса, в то время как атрибуты класса принадлежат самому классу. Различия заключаются в области видимости и способе доступа к атрибутам.

**8.** Инкапсуляция в Python: особенности, общедоступные (Public), не

общедоступные (Non-Public) и закрытые (Private) атрибуты.

Принцип универсального доступа, геттеры/сеттеры, свойства.

Ответ: Инкапсуляция в Python обеспечивается с помощью общедоступных (public), не общедоступных (non-public) и закрытых (private) атрибутов. Принцип универсального доступа может использоваться для безопасного доступа к закрытым атрибутам с использованием геттеров и сеттеров, а также свойств.

**9.** Наследование в Python: особенности, два вида наследования

(ключевые моменты, разница, примеры использования). Проверка принадлежности к классу/типу.

Ответ: Наследование в Python позволяет создавать новые классы на основе существующих. В Python поддерживаются как одиночное, так и множественное наследование.

**10.** Полиморфизм в Python: особенности, примеры использования.

Ответ: Полиморфизм в Python позволяет использовать один и тот же метод или функцию для объектов различных классов, что способствует повторному использованию кода и улучшению его читаемости.

**11.** Множественное наследование: возможности и недостатки. Использование класса Python как структуры из Си.

Ответ: Множественное наследование в Python может предоставить гибкость при проектировании классов, но также может усложнить иерархию классов и привести к конфликтам имен. Использование класса Python как структуры из Си может быть целесообразным для определения структур данных с фиксированным форматом памяти.