1. Что такое ООП?

ООП – объектно-ориентированное программирование. Это парадигма программирования, в которой все сущности представлены в виде классов и объектов – экземпляров класса. Объекты могут наследовать друг – друга.

1. Зачем люди придумали ООП?

Чтобы можно было строить гораздо более гибкие и масштабируемые системы, которые проще поддерживать, нежели те программы, которые написаны с использованием парадигмы функционального программирования.

1. Что такое класс?

Класс – сущность, обладающая свойствами (какими-то характеристиками) и методами – функциями, которые она может реализовывать.

1. Что такое объект (экземпляр класса)?

Объект – отдельно взятая сущность, обладающая свойствами и методами своего класса.

1. Что такое свойства класса?

Свойства класса – характеристики класса, описывающие сущность. Это свойства, которыми обладает класс и все объекты этого класса.

1. Что такое методы класса?

Методы класса – определённые функции, которые умеет реализовывать класс и все объекты этого класса.

1. Что такое модификаторы доступа (private, protected, public)?

Модификаторы доступа – позволяют задать видимость класса/метода/свойства. Private – видимость только внутри одного класса. Protected – внутри одного пакета. Public – видно везде.

1. Что такое интерфейс?

Интерфейс – это своего рода контракт, который указывает, что должен делать наш класс, который его имплементирует (реализует). Его часто путают с абстрактным классом, об их различиях будет в следующем вопросе. В некоторых языках между абстрактными классами ещё меньше отличий, например в Kotlin интерфейс может иметь реализацию методов, но это всё равно считается bad practice и общее соглашение требует соблюдать то, что интерфейсы ничего не реализуют, а только **указывают** на то, что мы обязательно должны реализовать. При реализации интерфейса нам нужно реализовать **все** методы, которые в нём описаны. И также интерфейс может иметь абстрактные свойства.

1. Что такое абстрактный класс?

Абстрактный класс – это класс, экземпляр которого невозможно создать. Это абстракция, которую проще описать на примере. Например, у нас есть абстрактный класс – транспортное средство. Transport. Каждое транспортное средство «умеет» ездить – имеет метод toRide. Следовательно, например классы Auto и Bike будут также транспортными средствами и будут переопределять метод toRide (реализация этого метода, кстати, в отличие от интерфейса уже может быть реализована как-то базово, в абстрактном классе). И также наш абстрактный класс может иметь свойства, которые могут быть тоже переопределены. Собственно, каждое наше транспортное средство имеет колёса в разном количестве – у авто их 4, у мотоцикла 2. Каждый из классов – Auto и Bike переопределяет свойство wheels. Это свойство абстрактное в абстрактном классе, ведь у каждого транспортного средства есть колёса, но не у каждого транспортного средства их 4. Ну и наследование – наследоваться можно лишь от одного класса, а интерфейсов реализовывать сколько угодно. Именно поэтому интерфейсы даже называют каким-либо действием, а абстрактный класс – существительным.

1. Что такое статический метод?

Статический метод – общий метод для всех классов, которые наследуют этот класс. Он может быть вызван без создания экземпляра класса. То есть статический метод привязывается к классу, а не к объекту, который создаётся когда-то потом позже.

1. Что такое инкапсуляция? Что такое наследование? Что такое полиморфизм?

Это три принципа объектно-ориентированного программирования.

Инкапсуляция – процесс обёртывания свойств/методов в классе в одно целое. Мы можем скрыть свойства/методы класса, и получать к ним доступ с помощью геттеров и сеттеров.

Наследование – помогает избегать дублирования кода и, если у нас есть какой-то класс, мы можем унаследоваться от него, и расширить наш класс – наследник, добавив в него свойства/методы. При этом мы можем пользоваться всеми свойствами/метода родителя.

Полиморфизм – свойство объекта, которое позволяет выполнять одно и то же действие разными способами. Например, в зависимости от входных данных.

Разница ООП Java и Python.

1. Изначально, если мы сами не запрещаем – в Python можно добавить в класс свойство, которое не было заранее объявлено. В Java так сделать нельзя.
2. В Java – best practice обозначать свойства класса приватными и далее управлять ими благодаря геттерам и сеттерам. В Python же – у нас по умолчанию всё публично, да и тут нет как такового private, только non-public, чтобы обозначить такую переменную, нужно добавить нижнее подчёркивание в начале.
3. В отличие от Java, нам нужно писать self (аналог this в Java). В Java его нужно писать только при одинаковом названии переменных, например, когда определяем сеттер - **this.a = a**. В Python же это обязательно.
4. Дальше – наследование. Python поддерживает множественное наследование, в отличие от Java.
5. В Java все классы, в том числе создаваемые пользователем – реализуют стандартные методы hashcode, equals, toString, потому что все неявно наследуются от класса Object. В Pytnon у нас предоставляются так называемые «магические методы» и их намного больше, чем дефолтных в Java. Среди них – init, eq, str, repr, add, lt…
6. Ну и множественное наследование – в Java мы можем наследоваться лишь от одного класса, а в Python – от нескольких.