**Object-Oriented Python**

**Q1. Does Python support OOP?**  
Yes, supports classes, objects, inheritance, polymorphism, encapsulation.

**Q2. What is self in Python classes?**  
Reference to the current object, similar to this in Java.

**Q3. Difference between class and instance variables?**  
Class variables shared by all objects, instance variables are per-object.

**Q4. Does Python support multiple inheritance?**  
Yes, unlike Java. Method Resolution Order (MRO) is followed.

**Q5. What are dunder (double underscore) methods?**  
 Special methods like \_\_init\_\_, \_\_str\_\_, \_\_len\_\_ that customize object behavior.

**Q6. What is \_\_init\_\_ in Python?**  
Constructor method called when an object is created.

**Q7. What is inheritance?**  
Reusing features of a parent class in a child class.

**Q8. Does Python support multiple inheritance?**  
Yes, handled via MRO (**Method Resolution Order**).

**Q9. What is encapsulation in Python?**  
 Restricting direct access using \_protected and \_\_private naming conventions.

**Q10. How does polymorphism work in Python?**  
Same method name can have different behavior depending on the object.

**Q11. What are abstract classes?**  
Classes with abstract methods (declared but not implemented). Use abc module.

**Q12. What is duck typing?**  
Behavior determined by methods/attributes, not type of object.

**Q13. Difference between classmethod, staticmethod, instancemethod?**  
- Instance: first arg self  
 - Class: first arg cls  
 - Static: no special first arg

**Q14. What is operator overloading?**  
Redefining operators using dunder methods like \_\_add\_\_, \_\_lt\_\_.

**Q15. Can Python classes be empty?**  
Yes, using pass.