1. OOPs (Object-Oriented Programming) – 10 Questions

1. What is the difference between self and cls in Python classes?

2. How does inheritance work in Python? Give an example with method overriding.

3. What is method overloading in Python? Is it supported natively?

4. Define constructor and destructor in Python. When are they called?

5. What is the difference between instance method, class method, and static method?

6. How do you restrict access to class attributes in Python (pseudo-private)?

7. Write a Python class to demonstrate encapsulation with getter/setter methods.

8. What is polymorphism in Python? Show it with two unrelated classes using the same method name.

9. What is a magic method? Name a few commonly used ones and their purpose.

10. How do you use isinstance() and issubclass() functions?

2. Decorators – 10 Questions

1. What is a decorator in Python and what is its typical use case?

2. Write a simple decorator that logs when a function is called.

3. Can you apply more than one decorator to a function? In what order are they applied?

4. What is the use of functools.wraps() in a decorator?

5. Convert the following decorator to one that accepts arguments (parameterized decorator).

6. How can you write a decorator to check if the user is logged in before accessing a function?

7. How does the @property decorator work? Give an example.

8. Write a decorator that catches and logs any exceptions in a function.

9. What is the difference between function decorator and class decorator?

10. Can decorators be used with class methods or static methods?

3. Generators – 10 Questions

1. What is a generator function? How is it different from a normal function?

2. Write a generator function to yield even numbers up to 20.

3. What happens if you call next() on an exhausted generator?

4. What is the use of yield? How does it help in memory efficiency?

5. How do you use a generator expression? How is it different from list comprehension?

6. Convert a normal function that returns a list into a generator.

7. How would you read a large file using a generator to process it line by line?

8. How does the generator maintain its state between calls?

9. What is the difference between return and yield inside a function?

10. What is the output of list(generator_function()) and how does it differ from a list-returning function?

4. Iterators – 10 Questions

1. What is the difference between an iterable and an iterator?

2. How do you make a class iterable using __iter__() and __next__()?

3. Explain what happens when StopIteration is raised.

4. Give an example of using iter() with a sentinel value.

5. How does a for loop work internally with iterators?

6. What built-in functions rely on iterators (e.g., map, zip, filter)?

7. How to manually loop over an iterator using next()?

8. Write a custom iterator that returns square of numbers from 1 to 5.

9. What happens when you try to iterate over an already exhausted iterator?

10. What is the use of the itertools module in python with itertools.