1. What are the new features added in Python 3.8 version?

Python 3.8 introduced several new features, including:

* Assignment expressions (walrus operator :=).
* Positional-only parameters in function definitions.
* The math.prod() function for calculating products of numbers.
* The functools.cached\_property decorator for creating cached properties.
* The typing.final decorator to indicate that a class or method is not meant to be subclassed or overridden.
* The importlib.metadata module for reading distribution metadata.
* The math.isqrt() function for integer square root.
* f-strings now support the = specifier for debugging and printing expressions.
* Syntax warning for using the backtick character.

1. What is monkey patching in Python?

Monkey patching in Python refers to dynamically modifying or extending existing modules or classes at runtime. It involves adding, modifying, or replacing methods, attributes, or functionality in existing code without altering its original source code. While monkey patching can provide flexibility, it can also lead to maintenance challenges and unexpected behavior.

1. What is the difference between a shallow copy and deep copy?

Shallow copy and deep copy are two ways of copying objects in Python:

* Shallow copy creates a new object, but it does not create new copies of nested objects. Changes to nested objects are reflected in both the original and the copied object.
* Deep copy creates a new object and recursively creates new copies of all nested objects. Changes to nested objects do not affect the original or other copies.

1. What is the maximum possible length of an identifier?

The maximum possible length of an identifier (variable, function, class name, etc.) in Python is implementation-dependent. In CPython, the most common implementation, identifiers can be up to 255 characters. However, it's considered good practice to keep identifiers reasonably short and meaningful for code readability.

1. What is generator comprehension?

Generator comprehension, similar to list comprehension, is a concise way to create generators in Python. It allows you to create and iterate over a generator object in a single line of code. It follows the syntax of [expression for item in iterable], where the generator yields values one at a time as the loop iterates over the iterable. Generator comprehensions are memory-efficient as they generate values on-the-fly, unlike list comprehensions that create an entire list in memory.