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

- Python 3.8 introduced several new features, including the "walrus operator" (`:=`), positional-only parameters, the `functools.cached\_property` decorator, f-strings improvements, assignment expressions, and more. It also included various performance enhancements and library updates. For a comprehensive list of features, you can refer to Python's official documentation or release notes.

2. What is monkey patching in Python?

- Monkey patching in Python refers to the practice of dynamically modifying or extending the behavior of classes, modules, or functions at runtime. It involves adding, modifying, or replacing attributes or methods in existing code, often to fix bugs or add functionality. While it can be useful, it should be used with caution as it can lead to unexpected behavior and compatibility issues.

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

- A shallow copy creates a new object but does not recursively duplicate the objects contained within the original object. In contrast, a deep copy creates a new object and recursively duplicates all objects, including nested objects, found within the original object. Changes to objects in a deep copy do not affect the original, whereas changes to objects in a shallow copy can impact the original.

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

- In Python, the maximum possible length of an identifier (variable name, function name, etc.) is not specified explicitly. However, it is practical to keep identifiers reasonably short and descriptive for readability. Extremely long identifiers may be less maintainable and go against Python's coding conventions.

5. What is generator comprehension?

- Generator comprehension is a concise way to create generator objects in Python. It uses a syntax similar to list comprehension but generates values lazily as needed, saving memory. Instead of creating a list, it creates a generator, which can be iterated over using a `for` loop or other iterable methods. Generator comprehensions use parentheses `( )` instead of square brackets `[ ]` like list comprehensions. For example:

```python

generator = (x for x in range(10))

```

This creates a generator that yields numbers from 0 to 9 when iterated over.