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

***Ans***:

1. f-strings support = for self-documenting expressions and debugging - This allows you to include expressions inside f-strings that will be evaluated and displayed alongside the string. This can be helpful for debugging or for self-documenting code.
2. More precise types - Python 3.8 introduces several new features related to type annotations, including the ability to specify types for variables in a way that allows for more precise type checking.
3. Simpler and faster debugging - Python 3.8 includes several improvements to the debugging experience, including a simplified traceback format and faster debugging of asyncio code.

4. Other changes - Other new features in Python 3.8 include improved support for Unix signals, new modules for working with dataclasses and asynchronous contexts, and several performance optimizations.

1. What is monkey patching in Python?

***Ans***:

Monkey patching is a technique in Python that allows you to modify the behaviour of an object or function at runtime by changing its attributes or methods. It is called monkey patching because it involves making changes to code on the fly, much like a monkey would tinker with objects in the real world.

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

***Ans***:

1. Shallow Copy: A shallow copy creates a new object that references the original object's memory location. It copies the references to the objects, rather than the objects themselves.

2. Deep Copy: A deep copy creates a new object with a new memory location and a new set of references to objects within the original object. It copies all objects recursively, so any changes made to the original object will not affect the copied object and vice versa.

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

***Ans***:

In Python, the maximum length of an identifier (variable name, function name, etc.) is technically unlimited. However, it is recommended to keep the length of identifiers reasonable and to use descriptive names that are easy to understand and remember.

5. What is generator comprehension?

***Ans***:

Generator comprehension is a concise and memory-efficient way of creating a generator object in Python. It is similar to list comprehension but instead of creating a list, it generates values on the fly as the generator is iterated.