Q1. What are the new features added in Python 3.8 version?

A1. Python 3.8, released in October 2019, introduced several new features and improvements. Here are some of the key features:

1. Assignment Expressions (The Walrus Operator): Python 3.8 introduced a new operator :=, also known as the "walrus operator", which allows us to assign and return a value in a single expression.
2. Positional-only Parameters: Python 3.8 introduced support for positional-only parameters in function definitions. This allows us to specify parameters that can only be passed positionally and not as keyword arguments.
3. f-strings now support the = operator for self-documenting expressions and debugging.
4. Python now uses the Load Time Calculation of CPython and there is a faster parser for Python.
5. The typing module now includes support for Protocols, which are used to define objects that satisfy certain interfaces or APIs.
6. Python 3.8 adds a new implementation of the math module, called math.isqrt, which calculates the integer square root of a number.
7. Python 3.8 introduces a new asyncio function called asyncio.all\_tasks, which returns a set of all currently pending tasks in an event loop.
8. The Python REPL now supports multiline editing and syntax highlighting.
9. Improvements to the subprocess module, including the ability to set timeouts on subprocess calls, and the ability to specify a list of accepted exit codes.

These are some of the key features introduced in Python 3.8.

Q2. What is monkey patching in Python?

A2.

Monkey patching in Python is a technique used to modify the behavior of a module, class, method, or function at runtime. It involves dynamically replacing, adding, or deleting attributes of an existing object to change its behavior. This can be useful for debugging or extending the functionality of existing code without modifying its source code. However, it can also be dangerous if not used carefully, as it can introduce unexpected behavior and make the code harder to debug and maintain. Therefore, it is generally recommended to use monkey patching sparingly and with caution.

Q3. What is the difference between a shallow copy and deep copy?

A3.

In Python, a shallow copy creates a new object but references the original elements, whereas a deep copy creates a new object and recursively copies all the elements from the original object.

In more detail, a shallow copy creates a new object but fills it with references to the same elements as the original object. This means that any changes made to the elements of the new object will also affect the original object. Shallow copies can be created using the **copy()** method or the slice operator **[:]**.

A deep copy creates a new object and recursively copies all the elements from the original object. This means that the new object is completely independent of the original object, and changes made to the new object will not affect the original object. Deep copies can be created using the **deepcopy()** method in the **copy** module.

Q4. What is the maximum possible length of an identifier?

A4.   
In Python, the maximum possible length of an identifier is not specified. It depends on the implementation of the Python interpreter being used. However, it is generally recommended to keep the identifier names short and descriptive, typically less than 79 characters in length.

Q5. What is generator comprehension?

A5.   
Generator comprehension is a concise way of creating a generator object in Python. It is a syntactic construct that allows us to generate elements on-the-fly as we iterate through them. Similar to list comprehension, generator comprehension uses a single line of code to generate a sequence of values, but instead of creating a list, it creates a generator object. This is particularly useful when working with large datasets as it doesn't create the entire sequence in memory at once, instead, it generates the values on-the-fly as they are needed, making it more memory-efficient. Additionally, because it is an iterator, it can only be iterated once, which is ideal for situations where we don't need to access the elements multiple times.