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

Python 3.8 introduced several new features and improvements. Here are some notable additions in Python 3.8:

* Assignment Expressions (The Walrus Operator): The introduction of the: = operator allows assignment expressions within other expressions. It allows you to assign a value to a variable as part of a larger expression, eliminating the need for a separate assignment statement.
* Positional-Only Parameters: Python 3.8 introduced support for defining positional-only parameters in function signatures. It allows specifying that certain function parameters can only be passed positional and not as keyword arguments.
* f-strings = Syntax: Python 3.8 introduced the = syntax in f-strings, which allows specifying the desired output formatting for variables within f-strings. It provides a convenient way to format variables and control their alignment, width, and precision.
* The math.prod () Function: Python 3.8 added the math.prod () function in the math module, which calculates the product of all elements in an iterable. It provides a concise way to compute the product without needing to write a loop explicitly.
* The math.isqrt () Function: The math.isqrt () function was introduced in Python 3.8. It returns the integer square root of a given number, rounded down to the nearest integer. It is useful for efficient integer square root calculations.
* Improved Syntax Warnings: Python 3.8 introduced more informative and user-friendly SyntaxWarning messages. These warnings provide clearer insights into potential syntax errors or problematic code constructs.
* Performance Improvements: Python 3.8 included various performance improvements and optimizations to make certain operations faster, such as dictionary lookups and concatenation of strings.

1. **What is monkey patching in Python?**

Monkey patching refers to the practice of modifying or extending existing code at runtime by adding, replacing, or modifying attributes or methods of objects. It involves making changes to the behaviour of existing classes or objects without modifying their original source code.

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

* Shallow Copy: A shallow copy creates a new object but references the same nested or referenced objects as the original. In other words, it copies the references to the nested objects rather than creating new copies of the nested objects themselves. Any changes made to the nested objects will be reflected in both the original and the shallow copy. Shallow copy can be achieved using the copy () method or the copy module in Python.
* Deep Copy: A deep copy, on the other hand, creates a completely independent copy of both the original object and any nested or referenced objects. It recursively copies all the objects, ensuring that the copied objects are completely separate from the original ones. Any modifications made to the nested objects in the deep copy will not affect the original. Deep copy can be achieved using the deepcopy () method or the copy module in Python.

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

In Python, the highest possible length of an identifier is 79 characters. Python is a high level programming language. It’s also a complex form and a collector of waste. Python, particularly when combined with identifiers, is case-sensitive. When writing or using identifiers in Python, it has a maximum of 79 characters. Unlikely, Python gives the identifiers unlimited length. However, the layout of PEP-8 prevents the user from breaking the rules and includes a 79-character limit.

1. **What is generator comprehension?**

A generator comprehension is a single-line specification for defining a generator in Python.

* It is absolutely essential to learn this syntax in order to write simple and readable code.
* Generator comprehension uses round bracket unlike square bracket in list comprehension.
* The generator yields one item at a time and generates item only when in demand. Whereas, in a list comprehension, Python reserves memory for the whole list. Thus we can say that the generator expressions are memory efficient than the lists.