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

ANSWER.

Python 3.8 introduced several new features and improvements over the previous versions. Some of the key features added in Python 3.8 include:

1. Assignment Expressions (The Walrus Operator):

- Python 3.8 introduced the `:=` operator, also known as the "walrus operator," which allows assignment expressions within other expressions.

- This operator assigns a value to a variable as part of an expression, reducing redundancy and improving readability.

2. Positional-Only Parameters:

- Python 3.8 introduced support for specifying positional-only parameters in function definitions using the `/` symbol.

- Parameters preceding the `/` symbol can only be passed positionally and cannot be specified using keyword arguments.

3. f-strings Support "=" for Self-documenting Expressions:

- Python 3.8 extended f-strings to support `=` for self-documenting expressions.

- With this feature, f-strings can include expressions along with their values, making the code more readable and self-explanatory.

4. TypedDict:

- Python 3.8 introduced the `TypedDict` class in the `typing` module, which allows the creation of dictionaries with a specific set of keys and value types.

- TypedDict provides static type checking support for dictionaries with predefined keys and value types.

5. Improved Syntax Warnings:

- Python 3.8 improved syntax warnings for various common syntax errors and code smells, helping developers write cleaner and more Pythonic code.

6. Performance Improvements:

- Python 3.8 included various performance improvements and optimizations, resulting in faster execution times and reduced memory usage for certain operations.

7. Additional Modules and Libraries:

- Python 3.8 introduced several new modules and libraries, including `importlib.metadata` for reading metadata from installed packages, `math.prod()` for calculating the product of a list of numbers, and `functools.cached\_property` for creating read-only cached properties.

1. What is monkey patching in Python?

ANSWER.

Monkey patching in Python refers to the practice of modifying or extending existing code or libraries at runtime by dynamically changing or replacing their attributes or methods. This is typically done by directly modifying the code or by injecting additional functionality into the existing codebase.

The term "monkey patching" comes from the idea of a monkey making changes to a piece of code on the fly, often in a haphazard or ad-hoc manner.

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

ANSWER.

The main difference between shallow copy and deep copy lies in how they handle nested objects within the original object. Shallow copy copies references to nested objects, while deep copy recursively creates independent copies of all nested objects. The choice between shallow copy and deep copy depends on the structure and complexity of the object being copied.

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

ANSWER.

In Python, the maximum possible length of an identifier (such as variable names, function names, class names, etc.) is implementation-defined. This means that the length limit can vary depending on the Python implementation being used (e.g., CPython, PyPy, Jython, IronPython, etc.) and the underlying platform or operating system.

1. What is generator comprehension?

ANSWER.

Generator comprehension, also known as generator expression, is a concise and memory-efficient way to create a generator in Python. It provides a compact syntax for generating elements lazily, one at a time, rather than creating a complete list upfront like list comprehensions.