**Assignment – 15**

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

Ans: Python 3.8 introduced several new features, including the walrus operator (:=), allowing assignment expressions within expressions; positional-only parameters, specifying that some function parameters must be passed positionally and cannot be passed by keyword; the typing module's support for TypedDict, providing a way to specify dictionaries with a fixed set of keys and value types; improved performance with the introduction of a new C API for thread-local storage (TLS); f-strings now support equal signs (=) to aid in debugging by showing the expressions' values; and various other improvements and optimizations.

1. What is monkey patching in Python?

Ans: Monkey patching in Python refers to the practice of dynamically modifying or extending classes or modules at runtime. This technique allows developers to change the behavior of code without modifying its original source. It involves replacing or adding methods or attributes to existing classes or modules, typically to fix bugs, add new features, or customize behavior to suit specific needs.

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

Ans: The difference between a shallow copy and a deep copy lies in how they handle nested objects. A shallow copy creates a new object but inserts references to the original nested objects. In contrast, a deep copy creates a new object and recursively copies all nested objects, resulting in completely independent copies. Shallow copies are faster and consume less memory, but changes to nested objects in one copy may affect other copies. Deep copies create fully independent copies but are slower and consume more memory, especially for deeply nested structures.

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

Ans: The maximum possible length of an identifier in Python is implementation-dependent. However, in practical terms, identifiers can be as long as memory allows, with a guideline of 255 characters being a common recommendation for readability and portability.

1. What is generator comprehension?

Ans: Generator comprehension, also known as generator expression, is a concise way to create a generator in Python. It follows a syntax similar to list comprehension but produces elements lazily, one at a time, as they are requested. Generator comprehensions use parentheses instead of square brackets and are useful for creating sequences without materializing the entire sequence in memory at once, which can be beneficial for memory efficiency and performance when dealing with large datasets or infinite sequences.