ASSIGNMENT 15

# Q1

Python 3.8 was released on October 14, 2019. It includes a number of new features, including:

Match expression: This is a new expression that allows us to match patterns in strings.

Literal collections: This allows us to create collections of literals, such as lists and sets, without having to use the list() or set() constructors.

Async/await syntax: This is a new syntax for writing asynchronous code.

New typing features: This includes new type annotations for union types, generics, and more.

Deque objects: This is a new data structure that allows us to efficiently insert and remove elements from the beginning or end of a list.

# Q2

Monkey patching refers to the practice of modifying or extending existing code at runtime, typically by adding, modifying, or replacing methods or attributes of an existing class or object. It allows developers to dynamically modify the behaviour of code without directly modifying the original source code.

# Q3

The difference between a shallow copy and a deep copy is related to how objects are copied in Python:

Shallow copy: A shallow copy creates a new object that references the original objects. The outer container is copied, but the inner objects are not duplicated. Changes made to the inner objects will be reflected in both the original and the shallow copy. In Python, we can use the copy() method or the copy module's copy() function to create a shallow copy.

Deep copy: A deep copy creates a new object with its own independent copy of all the objects found in the original. Both the outer container and the inner objects are duplicated, recursively creating new copies. Changes made to the original or the deep copy will not affect each other. In Python, we can use the copy module's deepcopy() function to create a deep copy.

# Q4

The maximum possible length of an identifier in Python is 79 characters. This is because identifiers are stored as strings in Python, and the maximum length of a string in Python is 79 characters.

# Q5

Generator comprehension, also known as generator expression, is a concise way to create a generator object in Python. It is similar to list comprehension or set comprehension, but instead of creating a list or set, it creates a generator.

A generator comprehension is defined within parentheses () and follows the syntax [expression for item in iterable if condition]. It allows us to generate values on-the-fly without storing them all in memory at once.