ASSIGNMENT - 15

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

Ans: Python 3.8 introduced several new features, including:

* Assignment expressions using the := operator (also known as the "walrus operator").
* Positional-only parameters in functions.
* The math.prod() function to calculate the product of a sequence.
* f-strings support = specifier for self-documenting expressions.
* functools.cached\_property for creating read-only properties.
* Syntax warning for = used for comparison instead of ==.
* New syntax features like the typing.Protocol class for structural subtyping.
* Performance improvements in dictionary handling.
* New built-in modules like importlib.metadata.

2. What is monkey patching in Python?

Ans: Monkey patching refers to the practice of dynamically modifying or extending the behavior of a class or module at runtime. It involves altering or adding attributes or methods to existing classes or modules during program execution. This technique can be used to modify behavior temporarily or to fix bugs without changing the original source code.

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

Ans: Shallow Copy vs. Deep Copy:

Shallow Copy: It creates a new object but inserts references to the objects found in the original. It copies the top-level structure of the object but not nested objects. The copy() method in Python's copy module or the copy() method of objects is used to create shallow copies.

Deep Copy: It creates a new object and recursively copies the objects found in the original. It duplicates the entire structure, including nested objects. The deepcopy() method in Python's copy module is used to perform deep copies.

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

Ans: In Python, the maximum possible length of an identifier (variable name, function name, etc.) is implementation-specific. There's no hard limit defined in the language specification, but practical limitations may exist based on the interpreter or implementation being used.

5. What is generator comprehension?

Ans: Generator comprehension, also known as generator expression, is a concise way to create generators in Python. It's similar to list comprehensions but generates values lazily, one at a time, instead of creating a full list in memory.

Example:

gen = (x for x in range(10) if x % 2 == 0) # Generator expression

for item in gen:

print(item)

This creates a generator that yields even numbers from 0 to 9 when iterated over, without storing the entire sequence in memory.