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

2. What is monkey patching in Python?

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

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

5. What is generator comprehension?

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

Python 3.8 introduced several new features and improvements. Some notable ones include:

**Assignment Expressions (Walrus Operator :=)**: Allows assignment within an expression using the syntax :=.  
python  
  
if (n := len(my\_list)) > 10:

print(f"List is too long ({n} elements)")

**Positional-Only Parameters**: Function parameters can be specified as positional-only using /.  
python  
  
def f(a, b, /, c, d, \*, e, f):

pass

**f-strings Support for Self-Documenting Expressions**: Added support to f-strings to include the expression text itself.  
python  
  
value = 42

print(f"{value=}") # Outputs: value=42

* **The typed Module**: Introduces the typing module to provide better support for type hints.
* **New Syntax Features**: Includes continue statements in finally blocks and the use of \_\_import\_\_ as a callable in Python's import system.
* **Shared Memory for Multiprocessing**: The multiprocessing module now has a shared\_memory class for sharing data between processes.
* **Improved Performance**: Various optimizations to improve the performance of Python.
* **Debugging and Development Enhancements**: New features like SyntaxWarning for incorrect syntax usage and new -X options to customize Python runtime behavior.

### **2. What is monkey patching in Python?**

Monkey patching refers to the practice of modifying or extending existing classes or modules at runtime. It involves altering or adding attributes or methods to classes or modules dynamically, usually to fix bugs or add functionality.

**Example**:

python

import some\_module

def new\_method(self):

return "New method!"

# Add or replace a method in an existing class

some\_module.SomeClass.new\_method = new\_method

# Now instances of SomeClass will have the new\_method

instance = some\_module.SomeClass()

print(instance.new\_method()) # Outputs: New method!

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

* **Shallow Copy**:
  + **Definition**: Creates a new object but inserts references into it to the objects found in the original. Changes to mutable objects within the copy will affect the original.
  + **Method**: copy.copy()

**Example**:  
python  
  
import copy

original = [[1, 2, 3], [4, 5, 6]]

shallow = copy.copy(original)

shallow[0][0] = 10

print(original) # Outputs: [[10, 2, 3], [4, 5, 6]]

* **Deep Copy**:
  + **Definition**: Creates a new object and recursively copies all objects found in the original, ensuring no references are shared.
  + **Method**: copy.deepcopy()

**Example**:  
python  
  
import copy

original = [[1, 2, 3], [4, 5, 6]]

deep = copy.deepcopy(original)

deep[0][0] = 10

print(original) # Outputs: [[1, 2, 3], [4, 5, 6]]

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

In Python, there is no explicit limit on the length of an identifier in the language specification. However, in practice, identifier lengths are limited by the constraints of the system or editor being used. The Python documentation does not specify a maximum length, but it is generally advisable to keep identifiers reasonably short for readability and maintainability.

### **5. What is generator comprehension?**

Generator comprehension is a concise way to create generators in Python, similar to list comprehensions but using parentheses instead of square brackets. It produces values one at a time and is memory-efficient for large datasets.

**Syntax**:

python

(expression for item in iterable if condition)

**Example**:

python

gen = (x \* x for x in range(10)) # Generator for squares of numbers from 0 to 9

for value in gen:

print(value)

In this example, gen is a generator that yields the squares of numbers from 0 to 9 one at a time.