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

#### **Assignment Expressions(:=)**

This operator is used to assign and return a value in the same expression. This removes the need for initializing the variable upfront. The major benefit of this is it saves some lines of code. It is also known as **“The Walrus Operator”**due to its similarity to the eyes and tusks of a walrus.

1. a **=** 20
2. **if** (b :**=** a) > 10:
3. print(f"The value of b is {b} and is greater than 10.")

#### **f-strings now support “=”**

This string formatting mechanism is known as Literal String Interpolation or more commonly as [F-strings](https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/) (because of the leading f character preceding the string literal). The idea behind f-strings is to make string interpolation simpler. Python 3.8 allows the use of the above-discussed assignment operator and equal sign (=) inside the f-strings.  
For example, say we have two variables “a” and “b”, and we want to print “a + b” along with the result. Here we can use f-strings=.

a **=** 5

b **=** 10

# Using = at the end of

# f-strings

**print**(f'{a + b = }')

# Using assignment operators

# inside f-strings

print(f'{(c := a + b)}')

print("The value of c:", c)

**Output:**

a + b = 15

15

The value of c: 15

#### **reversed() works with a dictionary**

Unlike Python 3.7, now in Python 3.8, the built-in method “reversed()” can be used for accessing the elements in the reverse order of insertion.  
**Example:**

# Declaring a dictionary

my\_dict **=** dict(x **=** 1, y **=** 2, z **=** 3)

# Prints only keys

list(reversed(my\_dict))

# Prints the key-value pair

# as a list of tuples

list(reversed(my\_dict.items()))

**op :**

['z', 'y', 'x']

[('z', 3), ('y', 2), ('x', 1)]

1. **What is monkey patching in Python?**

Monkey patching in python refers to **modifying or updating a piece of code or class or any module at the runtime**. In simple words, we can change the behavior or working of a class/ module at the runtime without changing the whole python code.

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

The Differences between a Shallow Copy and deep copy are as follows:

When an object is copied using **copy()**, it is called **shallow copy** as changes made in copied object will also make corresponding changes in original object, because both the objects will be referencing same address location.

When an object is copied using **deepcopy()**, it is called **deep copy** as changes made in copied object will not make corresponding changes in original object, because both the objects will not be referencing same address location.

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

 In Python, the highest possible length of an identifier is 79 characters. Python is a high level programming language. It’s also a complex form and a collector of waste.

* Python, particularly when combined with identifiers, is case-sensitive.
* When writing or using identifiers in Python, it has a maximum of 79 characters.
* Unlikely, Python gives the identifiers unlimited length.
* However, the layout of PEP-8 prevents the user from breaking the rules and includes a 79-character limit.

1. **What is generator comprehension?**

A generator comprehension is a single-line specification for defining a generator in Python.

* It is absolutely essential to learn this syntax in order to write simple and readable code.
* Generator comprehension uses round bracket unlike square bracket in list comprehension.
* The generator yields one item at a time and generates item only when in demand. Whereas, in a list comprehension, Python reserves memory for the whole list. Thus we can say that the generator expressions are memory efficient than the lists.