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

Ans. 1)The walrus operator (:=).

This allows you to assign variables inside an expression.

num = [1,2,3,4,5]

if( (size:=len(num)) < 10 ):

print(f”Length of list is small, size={size}”)

2) There is a new function parameter syntax (/) to highlight that some of the functions must be stated positionally and not by keyword arguments.

We also have an operator (\*) that indicates that the arguments must be keyword only.

def func( a,b,/,c,d,\*,e,f ):

print(a,b,c,d,e,f)

* a and b arguments are positional only.
* c and d arguments can be positional as well as keyword.
* e and f arguments are keyword only.

func(1,2, 3,4, e=5, f=6 ) #Valid - prints 1 2 3 4 5 6

func(1,2, c=3,d=4, e=5, f=6 ) #Valid - prints 1 2 3 4 5 6

### 3. PEP 590 ( Vectorcall)

This release has made some improvements in the vector call which is a fast calling protocol for CPython.

A new C API is introduced to optimize the calls of objects.

This feature was already used in CPython but with the new C API, “fastcall” convention can be used by a user-defined extension class.

### 4. PEP 574 ( Pickle Protocol 5 with out-of-band data)

Pickle is useful to transfer big amounts of data between Python processors to take full advantage of multicore processors.

It’s important to maximize the transfer speed by optimizing memory copies. Pickle protocol 5 now supports out-of-band data buffers and extra metadata is required.

* **PickleBuffer** type for **\_\_reduce\_ex\_\_** returns out-of-band buffers.
* **buffer\_callback** parameter while pickling handles out-of-band data buffers.
* **buffers** parameter while unpickling shows out-of-band data buffers.

### 5. F-strings now support = (Easy debugging)

A small improvement has been made in the f-strings formatting. They can now support = operator in f-strings that allows debugging easier.

A = 5

print(f’{A=}’) #prints A=5

### 6. Improved Typing

#### 6.1 PEP 591 ( Final qualifier )

Python now supports the “final”. Java programmers already know about this. It has 3 major uses:

* Declaring a class final will prevent it from inheriting.
* Declaring a variable final will prevent it from reassigning the value.
* Declaring a method final will prevent it from being overridden.

#### 6.2 PEP 586 (Literal types )

Literal types are useful to know the literal value of an attribute or a variable. They are useful in type checking.

Consider this expression:

0== False

This will give True as a result but 0 is of type integer and False is of type bool. So with literal, we can force type checks to be literally some specific type.

1. What is monkey patching in Python?

Ans. 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?

Ans.

| Shallow Copy | Deep Copy |
| --- | --- |
| Shallow Copy stores the references of objects to the original memory address. | Deep copy stores copies of the object’s value. |
| Shallow Copy reflects changes made to the new/copied object in the original object. | Deep copy doesn’t reflect changes made to the new/copied object in the original object. |
| Shallow Copy stores the copy of the original object and points the references to the objects. | Deep copy stores the copy of the original object and recursively copies the objects as well. |
| Shallow copy is faster. | Deep copy is comparatively slower. |

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

Ans. An identifier can have a maximum length of 79 characters in Python.

1. What is generator comprehension?

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

my\_list = [1, 3, 5, 9, 2, 6]

filtered\_gen = (item for item in my\_list if item > 3)

next(filtered\_gen)