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

Ans 1) The walrus operator ':='

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.

1. What is monkey patching in Python?

Ans In Python, the term monkey patch refers to dynamic (or run-time) modifications of a class or

module. In Python, we can actually change the behavior of code at run-time.

class A:

def func(self):

print ("func() is being called")

We use above class in below code and change behavior of func() at run-time by assigning different

value.

def monkey\_f(self):

print ("monkey\_f() is being called")

# replacing address of "func" with "monkey\_f"

A.func = monkey\_f

obj = A()

# calling function "func" whose address got replaced

# with function "monkey\_f()"

obj.func()

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

Ans A shallow copy means constructing a new collection object and then populating it with referenceto the child objects found in the original. In essence, a shallow copy is only one level deep The

copying process does not recurse and therefore won’t create copies of the child objects themselves.

A deep copy makes the copying process recursive. It means first constructing a new collection object and then recursively populating it with copies of the child objects found in the original. Copying an

object this way walks the whole object tree to create a fully independent clone of the original object and all of its children.

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

Ans 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?

Ans Generator Expressions or comprehension are somewhat similar to list comprehensions, but the

former doesn’t construct list object. Instead of creating a list and keeping the whole sequence in the

memory, the generator generates the next element in demand. When a normal function with a return

statement is called, it terminates whenever it gets a return statement. But a function with a yield

statement saves the state of the function and can be picked up from the same state, next time the

function is called The Generator Expression allows us to create a generator without the yield keyword.

#e.g:

gen=(i for i in range(0,10))

print(gen)

sum(gen)