Q-1: What is the difference between a function and a method in Python?

-> A function is an independent block of reusable code that is not associated with an object, whereas a method is a function that is associated with an object

-> EX: 1).

Function (independent) def greet(name): return f"Hello, {name}"

print(greet("Alice")) // Output: Hello, Alice

EX: 2).

Method (associated with an object) class Person: def **init**(self, name): self.name = name

def greet(self)  
    return f"Hello, {self.name}"

p = Person("Alice") print(p.greet()) # Output: Hello, Alice

Q-2:Explain the concept of function arguments and parameters in Python.

->Parameters are the variables listed in the function definition.

->Arguments are the actual values passed to the function when calling it.

EX: def add(a, b): return a + b

result = add(5, 3) print(result) // Output: 8

Q-3: What are the different ways to define and call a function in Python?

->#simple function

def square(n): return n \* n

print(square(4)) # Output: 16

-> # Function with lambda square = lambda n: n \* n

print(square(4)) # Output: 16

Q-4:What is the purpose of the return statement in a Python function?

->The return statement is used to send a value back to the caller from a function.

EX: def multiply(a, b):

return a \* b

result = multiply(3, 4)

print(result) # Output: 12

Q-5:What are iterators in Python and how do they differ from iterables?

->Iterable: An object that contains multiple values e.g., list, tuple, string.

->Iterator: An object that produces values from an iterable one at a time.

EX:

numbers = [1, 2, 3]

iterator = iter(numbers)

print(next(iterator)) # Output: 1

print(next(iterator)) # Output: 2

Q-6:Explain the concept of generators in Python and how they are defined.

->Generators are special functions that return values lazily using yield, instead of returning all values at once.

EX:

def count\_up\_to(n):

count = 1  
while count <= n:  
    yield count  
    count += 1

for num in count\_up\_to(5):

print(num)

Q-7:What are the advantages of using generators over regular functions?

->Memory efficient,Improves performance in loops.

Q-8:. What is a lambda function in Python and when is it typically used?

->A lambda function is an without a name that can have any number of arguments but only one expression.

EX:

square = lambda x: x \* x print(square(5)) # Output: 25

Q-9:Explain the purpose and usage of the map() function in Python

->The map() function applies a function to all elements in an iterable.

EX: numbers = [1, 2, 3, 4] squared = list(map(lambda x: x \* x, numbers)) print(squared) # Output: [1, 4, 9, 16]

Q-10:What is the difference between map(), reduce(), and filter() functions in Python

-> map(): Applies a function to each element in an iterable.

->filter(): Filters elements based on a condition.

->reduce(): Reduces an iterable to a single value.

EX: from functools import reduce

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

keyboard\_arrow\_down

map

squared = list(map(lambda x: x \* x, numbers)) print(squared) # Output: [1, 4, 9, 16, 25]

filter()

evens = list(filter(lambda x: x % 2 == 0, numbers)) print(evens) # Output: [2, 4]

reduce()

sum\_result = reduce(lambda x, y: x + y, numbers) print(sum\_result) # Output: 15