1. What is the relationship between def statements and lambda expressions ?

Ans: def statements are used to define named functions that can be called with arguments, while lambda expressions are used to create anonymous functions for use in contexts where a named function is not required and also it reduces the lines of code and statements.

2. What is the benefit of lambda?

Ans:

a. Requires less number of lines(Single line code)

b. lambda expressions are very flexible and can be used in a variety of contexts, including in list comprehensions, generator expressions, map() and filter() calls, and as arguments to other functions.

3. Compare and contrast map, filter, and reduce.

Ans:

Map : map requires when all elements inside iterator requires same requirements and iterator size won’t change.

Filter: filter requires when whole list elements need to get based on certain condition and iterator size can reduce compared to original iterator size.

Reduce: reduce will reduce the whole list elements into single element.

**Example:** addition of a list.

4. What are function annotations, and how are they used?

Ans: Function annotations are a Python feature that allows you to attach metadata to the parameters and return value of a function. They are optional and do not affect the behavior of the function itself.Function annotations are specified using colons (:) after the parameter or return value name, followed by an expression that specifies the annotation.

**Ex:**

def greet(name: str, age: int) -> str:

return f"Hello, {name}! You are {age} years old."

In this example, the annotations indicate that the name parameter should be a string, the age parameter should be an integer, and the return value should be a string.

5. What are recursive functions, and how are they used?

Ans: A recursive function is a function that calls itself one or more times within its own code. Recursive functions are used to solve problems that can be broken down into smaller subproblems that are similar in nature to the original problem.

**Ex:**

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

6. What are some general design guidelines for coding functions?

Ans:

a. Use 4-space indentation and no tabs.

b. Use docstrings

c. Wrap linethat they don’t exceed 79 characters

d. Use of regular and updated comments are valuable to both the coders and users

e. Use of trailing commas : in case of tuple -> ('good',)

f. Use Python’s default UTF-8 or ASCII encodings and not any fancy encodings

g. Naming Conventions

f. Characters that should not be used for identifiers : ‘l’ (lowercase letter el), ‘O’ (uppercase letter oh), ‘I’ (uppercase letter eye) as single character variable names as these are similar to the numerals one and zero.

h. Don’t use non-ASCII characters in identifiers

i. Name your classes and functions consistently

j. While naming of function of methods always use self for the first argument.

7. Name three or more ways that functions can communicate results to a caller.

Ans:

a. Return Statement

b. Global Variables

c. Side Effects

d. Exception Handling

e. Callback Functions

f. Yield Statement