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

Keyword def that marks the start of the function header. A function name to uniquely identify the function.

Function naming follows the same rules of writing identifiers in Python

Lambdas are one-line methods without a name or we can say anonymous Func

They work practically the same as any other method in Python Lambdas differ from normal Python methods because they can have only one expression, can't contain any statements and their return type is a function object

1. **What is the benefit of lambda?**

Fewer Lines of Code

Lambda functions are inline functions and thus execute comparatively faster Many times lambda functions make code much more readable by avoiding the logical jumps caused by function calls

1. **Compare and contrast map, filter, and reduce.**

map applies as a transformation to an element.

The map() function iterates through all items in the given iterable and executes the function we passed as an argument on each of them. Syntax : map(function, iterable(s))

filter accumulates only elements matching a condition.

filter() forms a new list that contains only elements that satisfy a certain condition, i.e. the function we passed returns True Syntax : filter(function, iterable(s))

reduce accumulates all elements to a single value, by using immutable values

1. **What are function annotations, and how are they used?**

Function annotation is the standard way to access the metadata with the arguments and the return value of the function.

These are nothing but some random and optional Python expressions that get allied to different parts of the function. They get evaluated only during the compile-time and have no significance during the run-time of the code. They do not have any significance or meaning associated with them until accessed by some third-party libraries. They are used to type check the functions by declaring the type of the parameters and the return value for the functions. The string-based annotations help us to improve the help messages.

Syntax :

def func(a: 'int') -> 'int':

pass

Annotations for simple parameters:

def func(x: 'float'=10.8, y: 'argument2'):

In the above code the argument, ‘x’ of the function func, has been annotated to float data type and the argument ‘y’ has a string-based annotation. The argument can also be assigned to a default value using a ‘=’ symbol followed by the default value. These default values are optional to the code.

Annotations for return values:

def func(a: expression) -> 'int':

The annotations for the return value is written after the ‘->’ symbol.

1. **What are recursive functions, and how are they used?**

A recursive function is a function that calls itself during its execution.

This means that the function will continue to call itself and repeat its behavior until some condition is met to return a result

1. **What are some general design guidelines for coding functions?**

Use 4-space indentation and no tabs.

Use docstrings

Wrap linethat they don’t exceed 79 characters

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

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

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

Naming Conventions

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

Function can return single value

Can return multiple values, tuple

can return list,dictionary

can return function object

can return class object