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

def statement is used to create a normal function. where as lamba expressions are used to create Anonymous functions. which can be assigned to a variable and can be called using the variable later in function.

Lambda's body is a single expression and not a block of statements like def statement.

Example:

def calculate\_cube\_root(x):

return x\*\*(1/3)

calculate\_cube\_root(27)

ouput:3.0

lamda:

# Define function using lambda for cube root

cube\_root= lambda x: x\*\*(1/3)

# Call the lambda function

print(cube\_root(27))

output:

3.0

|  |
| --- |
|  |

1. What is the benefit of lambda?

Can be used to create Nameless/Anonymous functions inside some complex functions if we are planning to use it only once.

Moderate to small functions can be created in a single line

Fuctions created using lambda expressions can be assigned to a variable and can be used by simply calling the variable

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

map(): The map() function is a type of higher-order. This function takes another function as a parameter along with a sequence of iterables and returns an output after applying the function to each iterable present in the sequence.

filter(): The filter() function is used to create an output list consisting of values for which the function returns true.

reduce(): The reduce() function, as the name describes, applies a given function to the iterables and returns a single value

**from** functools **import** reduce

*# map function*

print('Map ->',list(map(**lambda** x:x**+**x, [1,2,3,4])))

*# fitler function*

print('Filter ->',list(filter(**lambda** x:x**%2** !=0, [1,2,3,4])))

*# reduce function*

print('Reduce ->',reduce(**lambda** x,y:x**+**y, [1,2,3,4,5,6]))

Map -> [2, 4, 6, 8]

Filter -> [1, 3]

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

Function annotations provide a way of associating various parts of a function with arbitrary pythoncexpressions at compile time.  
Annotations of simple parameters def func(x: expression, y: expression = 20):  
Whereas the annotations for excess parameters are as − def func (\*\*args: expression, \*\*kwargs: expression):

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

The term [Recursion](https://www.geeksforgeeks.org/recursion/#:~:text=The%20process%20in%20which%20a,%20can%20be%20solved%20quite%20easily.) can be defined as the process of defining something in terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.

Advantages:

A complicated function can be split down into smaller sub-problems utilizing recursion.

Sequence creation is simpler through recursion than utilizing any nested iteration.

Recursive functions render the code look simple and effective.

Disadvantages :

A lot of memory and time is taken through recursive calls which makes it expensive for use.

Recursive functions are challenging to debug.

The reasoning behind recursion can sometimes be tough to think through.

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

Always try to write code in try and except block.

Always try to make log file for the code.

Always use a docstring to explain the functionality of the function

avoid using or limited use of global variables

Proper Identation to increase the code readability

try to follow a naming convention for function names (pascalCase or camelCase) and stick with the same convention throughout the application.

Avoid using digits while choosing a variable name

try to use a name for the function which conveys the purpose of the function

Local variables should be named using camelCase format (ex: localVariable) whereas Global variables names should be using PascalCase (ex:GlobalVariable).

Constant should be represented in allcaps (ex:CONSTANT).

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

print

return

yield