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

The def functions must be declared in the namespace. The def functions can perform any python task including multiple conditions, nested conditions or loops of any level, printing, importing libraries, raising Exceptions, etc. Lambda function must returns object , The [lambda functions](https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/) can be used without any declaration in the namespace. The lambda functions defined above are like single-line functions. These functions do not have parenthesis like the def defined functions but instead, take parameters after the lambda keyword as shown above. There is no return keyword defined explicitly because the lambda function does return an object by default. Eg : a = lambda \*a : a\*a

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

No need to have return statement and no need to provide name for the function. No of lines can be reeduced.To write a simple function with shorter code Lambda can be used.

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

**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))

numbers = (1, 2, 3, 4)

result = map(lambda x: x + x, numbers)

print(list(result))

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

**The filter() function** filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

# a list contains both even and odd numbers.

seq = [0, 1, 2, 3, 5, 8, 13]

# result contains odd numbers of the list

result = filter(lambda x: x % 2 != 0, seq)

print(list(result))

# result contains even numbers of the list

result = filter(lambda x: x % 2 == 0, seq)

print(list(result))

Res -> Odd -> [1, 3, 5, 13]

Res -> Even - > [0, 2, 8]

**The reduce(fun,seq)** function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.This function is defined in “functools” module.

Working :

At first step, first two elements of sequence are picked and the result is obtained.

Next step is to apply the same function to the previously attained result and the number just succeeding the second element and the result is again stored.

This process continues till no more elements are left in the container.

The final returned result is returned and printed on console.

 # importing functools for reduce()

import functools

lis = [1, 3, 5, 6, 2, ]

print("The sum of the list elements is : ", end="")

print(functools.reduce(lambda a, b: a+b, lis))

 print("The maximum element of the list is : ", end="")

print(functools.reduce(lambda a, b: a if a > b else b, lis))

Res -> The sum of the list elements is : 17

The maximum element of the list is : 6

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

The benefits from function annotations can only be reaped via third party libraries. The type of benefits depends upon the type of the library.  The function annotations in the above code can be accessed by a special attribute ‘\_\_annotations\_\_’.

Ex :

[def foo(a:”int”, b:”float”=5.0) -> ”int”]

def fib(n:'int', output:'list'=[])-> 'list':

print(foo.\_\_annotations\_\_)

print(fib.\_\_annotations\_\_)

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

Function that call itself is called recursive Fn .

Ex :

def recursive():

<<Any Code>>

recursive()

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

Code should be Reusable , To handle Exceptions properly , low latency to complete the function , logging is enabled to trouble shoot the issue.

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

return , Yield , write data to file , update global variable, break , call the caller function in exception handling part.