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** body is a single expression and not a block of statements like **def** statement. The lambda expression's body is similar to what we put def body's return statement. We simply type the result as an expression instead of explicitly returning it.

1. What is the benefit of lambda?

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

reduce(): function applies a given function to the iterables and returns a single value

# 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

from functools import reduce

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

**Output**

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

Filter --> [1, 3]

Reduce --> 21

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

Function annotations provide a way of associating various parts of a function with arbitrary python expressions at compile time.

Annotations of simple parameters

def func(x: expression, y: expression = 20):

Whereas the annotations for excess parameters is

def func (\*\*args: expression, \*\*kwargs: expression):

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

A recursive function is a function that calls itself during its execution. The process may repeat several times, outputting the result and the end of each iteration.

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

* 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).

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

1. print
2. return
3. yield