1)what is the relationship between def statements & lambda expression ?

Ans :def defined functions are commonly used because of their simplicity. The def defined functions do not return anything if not explicitly returned whereas the lambda function does return an object. 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.

Syntax : def function\_name:

function definition statements.

The lambda functions can be used without any declaration in the namespace. The lambda functions defined single-line functions. These functions do not have parenthesis like the def defined functions but instead, take parameters after the lambda keyword .There is no return keyword defined explicitly because the lambda function does return an object by default.

Syntax : lambda arguments : expression

Ex : x = lambda a : a + 10

print(x(5))

2)what is the benefits of lamda ?

Ans : The following are some of the benefits of lambda expressions:

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

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

3)compare & contrast map,filter,& reduce

Ans :MAP function to each element of iterator and collects result.

FILTER applies function to each element of iterator and collect those elements for which function returns true. For this function need to return boolean value.

REDUCE applies rolling computation to sequential pair of elements in iterator. Initially it takes two elements from iterator in sequence, applies function, collect result, then take next element in sequence from iterator, applies function and repeat this until list has single value.

>> a=[1,2,3,4]

>>> b = list(map(lambda x:x\*2, a))

>>> b

[2, 4, 6, 8]

>>> m = list(filter(lambda x:x>2, a))

>>> m

[3, 4]

>>> from functools import reduce

>>> p = reduce(add, a)

>>>

10

4)what are function annotations and how they are used ?

Ans : 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 are as − def func (\*\*args: expression, \*\*kwargs: expression):

5)what are recursive function and how are they used ?

Ans : : 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.

6)what are some General design guidelines for coding functions ?

Ans : Some of the general design guidelines for coding functions are:

Always use a docstring to explain the functionality of the function

avoid using or limited use of global variables

Proper Indentation 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 function can communicate result to a caller .

Ans : Some of the ways in which a function can communicate with the calling function is:

print

return

yield