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

:-

Both def statements and lambda expressions are used to create functions in Python.

A def statement is used to define a named function that can have any number of arguments and a body of statements that are executed when the function is called.

A lambda expression creates an anonymous function that can have any number of arguments but can only have a single expression in its body. It returns a function object that can be assigned to a variable or passed as an argument to another function.

2. What is the benefit of lambda?

:-

Lambda functions are commonly used as arguments to higher-order functions or used as a shorthand way to define simple functions.

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

:-

map() applies a function to each element of an iterable and returns a new iterable with the results. The returned iterable has the same length as the original iterable.

lst = [1, 2, 3]; squared = map(lambda x: x\*\*2, lst)

filter() applies a function to each element of an iterable and returns a new iterable with the elements for which the function returns True. The returned iterable may have fewer elements than the original iterable.

lst = [1, 2, 3, 4, 5]; even = filter(lambda x: x % 2 == 0, lst)

reduce() applies a function to the first two elements of an iterable, then applies the function to the result and the next element, and so on, until all elements have been processed. The final result is a single value.

from functools import reduce

lst = [1, 2, 3, 4, 5]; product = reduce(lambda x, y: x\*y, lst)

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

:-

Function annotations are a feature in Python that allow developers to add metadata or annotations to function parameters and return values. Annotations can be any expression and are specified using a colon after the parameter or return value name, followed by the expression.

example:

def calculate\_tax(income: float, rate: float = 0.2) -> float:

"""Calculate the tax owed on a given income at a given tax rate."""

return income \* rate

In this example, the annotations float are used to indicate that the income and rate parameters should be floating-point numbers, and the return value should also be a floating-point number.

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

:-

A recursive function is a function that calls itself during its execution. This technique is known as recursion, and it allows a function to solve a problem by breaking it down into smaller subproblems.

A recursive function has two parts: a base case and a recursive case. The base case is the condition that stops the recursion and returns a value. The recursive case is the condition that calls the function again, passing in new arguments, until the base case is met.

def factorial(n):

if n == 1:

return 1

else:

return n \* factorial(n-1)

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

:-

Function names should be descriptive and should clearly indicate what the function does.

Functions should be small and modular.

Functions should have a clear input and output.

Functions should avoid global variables

Functions should handle errors gracefully and should not crash the program.

Functions should be written with performance in mind, but not at the expense of readability or maintainability.

Functions should follow a consistent coding style and should be well documented.

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

:-

Return statement: Functions can return a value using the return statement. This allows the caller to receive and use the value returned by the function.

Global variables: Functions can also modify global variables, which can be accessed by the caller after the function has completed execution.

Side effects: Functions can modify non-global variables, objects, or files, which can be accessed by the caller after the function has completed execution. These changes are known as side effects.