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

Both def statements and lambda expressions are used to define functions in Python. The key difference is that def statements define named functions with a block of code, while lambda expressions define anonymous functions using a concise expression. Lambda expressions are often used for simple and short operations, while def statements are used for more complex functions.

2. What is the benefit of lambda?

Lambda expressions offer a concise way to create small, throwaway functions for simple operations. They are often used when a function is needed as an argument for another function (like in map, filter, etc.) or when writing short, one-liner functions.

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

* map: Applies a function to all elements in a sequence and returns an iterator of the results.
* filter: Applies a function to elements in a sequence and returns an iterator of elements that satisfy a condition.
* reduce: Applies a function cumulatively to elements in a sequence and returns a single value.

All three functions operate on sequences, but they differ in their purpose and output. map transforms all elements, filter selects elements based on a condition, and reduce aggregates elements into a single value.

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

Function annotations are optional metadata that can be added to the parameters and return value of a function. They provide additional information about the expected types of inputs and outputs. Annotations are defined using colons after the parameter names and after the closing parenthesis for the return value. For example:

def add(a: int, b: int) -> int:

return a + b

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

Recursive functions are functions that call themselves as part of their execution. They are used to solve problems that can be broken down into simpler subproblems of the same nature. Recursive functions typically have a base case that defines when the recursion should stop and one or more recursive cases that call the function with smaller inputs.

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

Functions should have a clear and descriptive name.

Functions should do one thing and do it well (single responsibility principle).

Use meaningful parameter names and follow naming conventions.

Keep functions short and focused.

Avoid global variables inside functions.

Include comments and docstrings to explain the purpose and usage of the function.

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

* Return values: Functions can use the return statement to provide a result back to the caller.
* Modifying mutable objects: Functions can modify mutable objects (like lists or dictionaries) that are passed as arguments, and the modifications will be visible outside the function.
* Global variables: Functions can read and modify global variables, although it's generally recommended to avoid this for better code organization.
* Function annotations: Annotations can provide additional information about the expected return value and parameter types.
* Printing: Functions can print results directly to the console for the user to see.