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

Relationship between def statements and lambda expressions:

* Both def statements and lambda expressions are used to create functions in Python.
* def statements are used for creating named functions with multiple expressions, statements, and a specified block of code.
* lambda expressions are used for creating anonymous functions (functions without a name) with a single expression.

1. What is the benefit of lambda?

Benefit of lambda:

* The main benefit of lambda expressions is conciseness and readability in creating small, one-line functions.
* They are often used in situations where a short-lived, simple function is needed, such as in functional programming constructs like map, filter, and sorted.

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

map:

Applies a given function to all items in an iterable and returns an iterator of the results.

Syntax: map(function, iterable)

filter:

Filters items in an iterable based on a given function's criteria and returns an iterator of the items that meet the criteria.

Syntax: filter(function, iterable)

reduce:

Successively applies a function to the items in an iterable to reduce them to a single accumulated result.

Requires importing from the functools module in Python 3.

Syntax: functools.reduce(function, iterable[, initializer])

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

Function annotations are a way to attach metadata to the parameters and return value of a function. They are expressed as arbitrary expressions and do not affect the function's behaviour.

Annotations are often used for documenting the expected types of function arguments or the intended purpose of the function.

Ex:

def add(x: int, y: int) -> int:

return x + y

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

Recursive functions are functions that call themselves either directly or indirectly to solve a problem. They involve breaking down a problem into smaller instances of the same problem until a base case is reached.

Ex:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

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

Functions should have a clear purpose and perform a single task.

Use meaningful names for functions and variables.

Follow the principle of modularity by breaking down complex tasks into smaller, manageable functions.

Functions should have a consistent and predictable behavior.

Avoid side effects in functions unless clearly documented.

Provide clear and concise documentation (docstrings) for functions.

Ensure functions are appropriately tested.

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

Return statement: Functions can use the return statement to send a value back to the caller.

Print statements: Functions can print information for debugging purposes, but printing is not a recommended way to communicate results.

Global variables: Functions can modify global variables to communicate information (not recommended for modularity).

Exception raising: Functions can raise exceptions to indicate errors or exceptional conditions.

Function annotations: Annotations can be used to indicate the expected types of arguments and return values.