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

Ans1

Both def statements and lambda expressions are used to define functions. They have some differences in terms of syntax, behavior, and use cases.

A def statement is used to define a named function with a block of code that executes when the function is called.

def function\_name(arguments):

return value

A lambda expression, on the other hand, is used to create an anonymous function with a single expression that is evaluated and returned when the function is called.

lambda arguments: expression

2. What is the benefit of lambda?

Ans2

nefit of lambda?

Lambda expressions have several benefits

Concise syntax: Lambda expressions have a very concise syntax, which allows to define a function in a single line of code. This makes them ideal for use in situations where need a simple function, but don't want to define a full named function.

Anonymous functions: Lambda expressions are anonymous, which means that they don't need a name to be defined. This makes them ideal for use in situations where only need to define a function once and don't want to clutter up code with unnecessary function names.

Functional programming: Lambda expressions are commonly used in functional programming, which is a programming paradigm that emphasizes the use of functions as the primary building blocks of a program.

Readable code: Although lambda expressions can sometimes be difficult to read, they can also make code more readable in some situations.

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

Ans3

map(), filter(), and reduce() are three built-in functions.

map(): The map() function applies a given function to each item of an iterable (such as a list) and returns a new iterable with the results. The syntax of the map() function is as follows

map(function, iterable)

filter(): The filter() function applies a given function to each item of an iterable and returns a new iterable containing only the items for which the function returns True. The syntax of the filter() function is as follows

filter(function, iterable)

reduce(): The reduce() function applies a given function to the items of an iterable in a cumulative way and returns a single value. The syntax of the reduce() function is as follows

reduce(function, iterable)

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

Ans4

Function annotations are a feature that allow to add metadata to the arguments and return value of a function. Function annotations are defined using a colon followed by an expression that describes the type or other properties of the argument or return value.

def greet(name: str, age: int) -> str:

return f"Hello, {name}! You are {age} years old."

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

Ans5

Recursive functions are functions that call themselves as part of their execution. In other words, a recursive function is a function that solves a problem by breaking it down into smaller sub problems and solving each sub problem by applying the same function to it recursively. Recursive functions can be used to solve problems that can be expressed in terms of smaller versions of the same problem.

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

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

Ans6

some general design guidelines for coding functions

Keep functions short and focused.

Use descriptive and concise names for functions.

Make sure functions have a clear purpose and return value. Functions should have a well-defined purpose and should return a value that reflects that purpose.

Avoid side effects. Side effects are changes to the state of the program or system outside of the function's scope. They can make code harder to reason about and can cause unexpected behavior.

Use default values and optional arguments when appropriate. Default values and optional arguments can make functions more flexible and easier to use. They can also help to reduce the amount of repetitive code.

Avoid using global variables. Global variables can make it harder to reason about a program's state and can cause unexpected behavior.

Document functions clearly. Functions should have clear and concise documentation that explains what the function does, what its arguments are, and what its return value is.

Use error handling and raise meaningful exceptions. Functions should handle errors and exceptions gracefully, and raise exceptions with clear and meaningful messages when necessary.

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

Ans7

Return values: Functions can return values to the caller using the return statement. The return value can be a single value or a collection of values, such as a tuple or a list.

Output arguments: Functions can modify the value of one or more arguments that are passed in as references, rather than returning a value.

Exceptions: Functions can signal errors or other exceptional conditions by raising exceptions. The caller can then catch and handle the exception as needed.