**Yashwant Desai – Assignment 24**

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

Answer: Both def statements and lambda expressions are used for defining functions but they serve different purposes based on their characteristics and use cases. Def functions are more versatile and allow for complex logic, multiple statements, and better readability due to their named nature. Lambda expressions are more concise and are often used when you need a small function for a short and straightforward task especially in scenarios where you would pass a function as an argument to another function.

Def statement:



Lambda expression:



2. What is the benefit of lambda?

Answer: for certain tasks using lambda expressions can help reduce the complexity of the code by eliminating the need to define a separate named function for a simple operation. Lambda expressions allow us to define small, simple functions in a compact and concise manner. Lambda expressions are anonymous functions, meaning they don't need a formal name.

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

Answer: The map is used to transform each element of a sequence based on some function and get a new iterable with the transformed values. The filter is used to selectively extract elements from a sequence based on a condition defined by the function. The reduce is used when you need to perform cumulative calculations or aggregations on elements of a sequence.

Map: Applies a function to each element and returns an iterable with the results.

Filter: Selectively filters elements based on a given function's condition.

Reduce: Cumulatively performs an operation on sequential pairs of elements, reducing them to a single value.

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

Answer: Function annotations are specified using a colon (:) following the parameter or return type. Function annotations are primarily used for documentation purposes to provide hints about the function's intended usage and the expected data types.



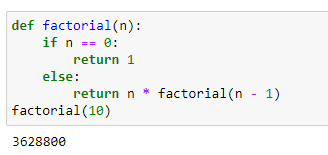
5. What are recursive functions, and how are they used?

Answer: Recursive functions are functions that call themselves to solve a problem. Recursive function in Python has two parts.

1 Base Case - This helps us to terminate the recursive function. It is a simple case that can be answered directly and doesn't use recursion. If satisfied, it returns the final computable answer. If this is omitted, the function will run till infinity.

2 General /Recursive Case - This case uses recursion and is called unless the base condition is satisfied.

Below is an example of recursive function

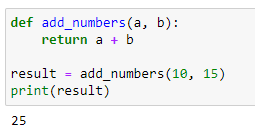


6. What are some general design guidelines for coding functions?

Answer: General design guidelines for coding functions are; each function should have a clear purpose and should ideally perform only one specific task. Use descriptive and meaningful names for your functions. Choose appropriate parameter names. Handle errors gracefully using try-except blocks.

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

Answer: Functions can communicate results to a caller in various ways. The common methods are: 1 Return Statement 2 Using Global Variables 3 Modifying Mutable Objects. The return statement is the most common and preferred way to communicate results in Python functions. Below is an example.



**Regards,**

**Yashwant**