Python (Functions Assignment) – 1 Solutions

Submitted by: Sweta Dhara

1. What is a lambda function in Python, and how does it differ from a regular function?

A: A lambda function is an anonymous function (i.e., defined without a name) that can take any number of arguments but, unlike normal functions, evaluates and returns only one expression. The [lambda functions](https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/) can be used without any declaration in the namespace. The lambda functions defined above are like single-line functions. These functions do not have parenthesis like the def defined functions but instead, take parameters after the lambda keyword as shown above. There is no return keyword defined explicitly because the lambda function does return an object by default.

Lambda functions and regular functions in Python differ in several aspects:

Syntax:

Lambda functions are defined using the lambda keyword followed by a list of parameters and an expression. They are typically used for simple, one-liner functions.

Regular functions are defined using the def keyword followed by a function name, a list of parameters, and a block of code. They can be used for more complex functions that require multiple lines of code.

Function Name:

Lambda functions are anonymous functions, meaning they don't have a proper name. They are often used for short-lived and single-use scenarios.

Regular functions have a proper name, allowing them to be called, reused, and referenced from various parts of the code.

Number of Expressions:

Lambda functions can only consist of a single expression, which is evaluated and returned as the function's result.

Regular functions can contain multiple statements and expressions. They allow you to have more complex logic within the function body.

Return Statement:

Lambda functions automatically return the result of the expression without requiring an explicit return statement.

Regular functions use the return statement to specify the value to be returned explicitly.

Usage:

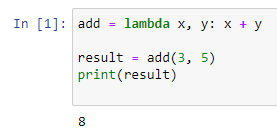
Lambda functions are typically used for short, simple operations where a small function is required, such as in functional programming with map(), filter(), and sorted().

Regular functions are more versatile and can be used for any kind of operation, regardless of complexity. They are suitable for code reusability and creating well-organized programs.

2. Can a lambda function in Python have multiple arguments? If yes, how can you define and use

them?

A: Yes, a lambda function in Python can have multiple arguments. Example:

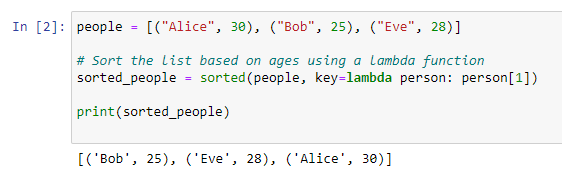


3. How are lambda functions typically used in Python? Provide an example use case.

A: Lambda functions in Python are often used for short, simple operations where you don't want to define a full named function using the def keyword. Lambda functions are useful for small tasks that are not reused throughout your code. They are commonly used in scenarios where you need to pass a function as an argument to another function (higher-order functions) or when you need to define a quick, throwaway function for a specific task.

Here's an example use case to illustrate how lambda functions are used:

Suppose you have a list of tuples containing names and corresponding ages, and you want to sort the list based on the ages. You can achieve this using the sorted function with a lambda function as the key argument:



In this example, the lambda function lambda person: person[1] is used as the key argument to the sorted function. It instructs sorted to sort the list of people based on the second element (age) of each tuple.

4. What are the advantages and limitations of lambda functions compared to regular functions in

Python?

A: Advantages of Lambda Functions:

Conciseness: Lambda functions are more concise and can be defined in a single line. This makes them suitable for simple operations where a full named function would be overkill.

Readability: For very simple operations, lambda functions can sometimes be more readable due to their inline nature. They keep the operation close to where it's being used.

Functional Programming: Lambda functions are commonly used in functional programming paradigms, where you pass functions as arguments to higher-order functions like map, filter, and reduce.

Anonymous Functions: Lambda functions are anonymous by nature, which means you don't need to come up with a function name. This can be useful for throwaway functions.

Limitations of Lambda Functions:

Single Expression: Lambda functions can only contain a single expression. This means you can't have multiple statements or complex logic within a lambda.

No Statements: You can't include statements like if, for, or while loops inside a lambda function. Only expressions are allowed.

Limited Readability: While lambda functions can be more readable for simple operations, they can become less readable as the complexity of the operation increases. Named functions can have meaningful names that convey their purpose.

No Documentation Strings: Lambda functions can't have docstrings, which are essential for providing documentation about the function's purpose and usage.

Reduced Reusability: Lambda functions are best suited for one-time or very specific use cases. For functions that you plan to reuse or that have complex logic, regular named functions are more appropriate.

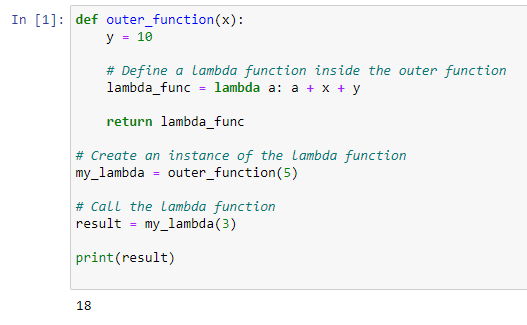
Limited Debugging: Debugging lambda functions can be more challenging due to their lack of a meaningful name and the inability to include print statements or breakpoints directly within them.

Top of Form

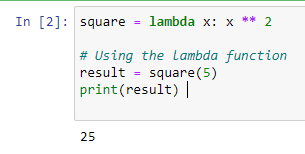
5. Are lambda functions in Python able to access variables defined outside of their own scope?

Explain with an example.

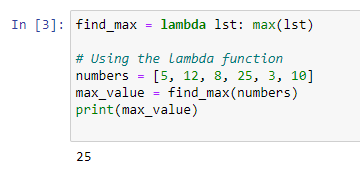
A: Yes, lambda functions in Python can access variables that are defined outside of their own scope. They have access to variables from the surrounding scope where they are created, just like regular functions do. Example:



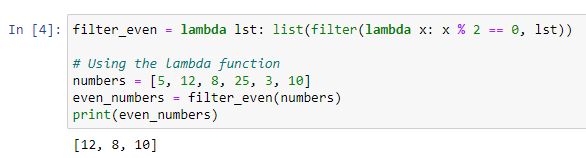
6. Write a lambda function to calculate the square of a given number.

A: 

7. Create a lambda function to find the maximum value in a list of integers.

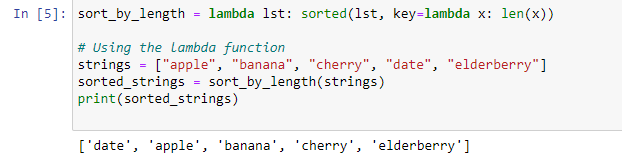
A: 

8. Implement a lambda function to filter out all the even numbers from a list of integers.

A: 

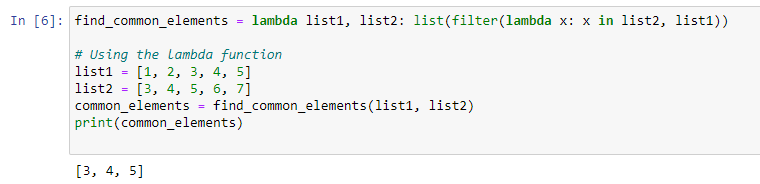
9. Write a lambda function to sort a list of strings in ascending order based on the length of each

string.

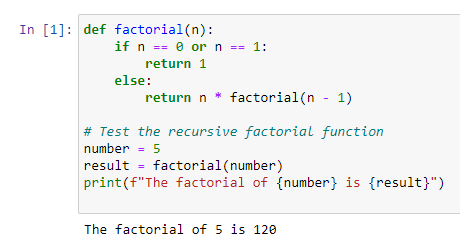
A: 

10. Create a lambda function that takes two lists as input and returns a new list containing the

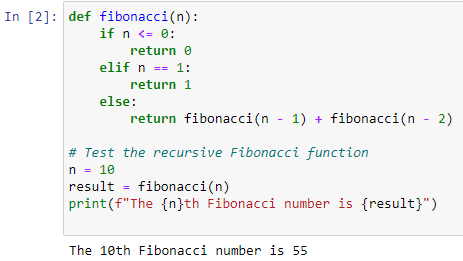
common elements between the two lists.

A: 

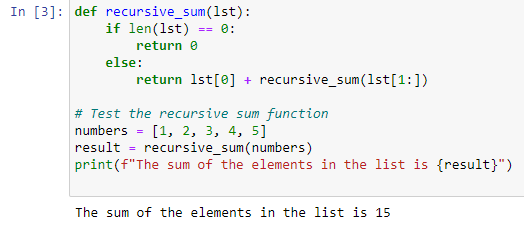
11. Write a recursive function to calculate the factorial of a given positive integer.

A: 

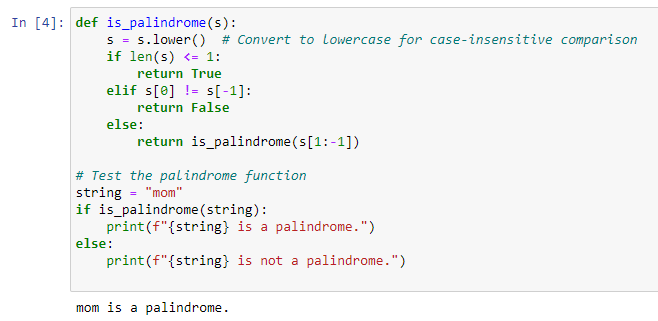
12. Implement a recursive function to compute the nth Fibonacci number.

A: 

13. Create a recursive function to find the sum of all the elements in a given list.

A: 

14. Write a recursive function to determine whether a given string is a palindrome.

A: 

15. Implement a recursive function to find the greatest common divisor (GCD) of two positive integers.

A: 