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

Answer: def defined functions are commonly used because of their simplicity. The def defined functions do not return anything if not explicitly returned whereas the lambda function does return an object. The def functions must be declared in the namespace. The def functions can perform any python task including multiple conditions, nested conditions or loops of any level, printing, importing libraries, raising Exceptions, etc.

2. lambda expression = The lambda functions 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.

2. What is the benefit of lambda?

Answer:
The lambda keyword in Python provides a shortcut for declaring small anonymous functions. Lambda functions behave just like regular functions declared with the def keyword. They can be used whenever function objects are required.

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

Answer:

The functions map(), filter(), and reduce() all do the same thing: They each take a function and a list of elements, and then return the result of applying the function to each element in the list. As previously stated, Python has built-in functions like map(), filter(), and reduce()

map creates a new array by transforming every element in an array individually. filter creates a new array by removing elements that don't belong. reduce , on the other hand, takes all of the elements in an array and reduces them into a single value. Just like map and filter , reduce is defined on Array

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

Answer: Function annotations are completely optional both for parameters and return value. Function annotations provide a way of associating various parts of a function with arbitrary python expressions at compile time. Annotations have a number of uses, among them: Information for the compiler — Annotations can be used by the compiler to detect errors or suppress warnings. Compile-time and deployment-time processing — Software tools can process annotation information to generate code, XML files, and so forth.

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

A recursive function is a function that calls itself during its execution. The process may repeat several times, outputting the result and the end of each iteration.

The function Count() below uses recursion to count from any number between 1 and 9, to the number 10. For example, Count(1) would return 2,3,4,5,6,7,8,9,10. Count(7) would return 8,9,10. The result could be used as a roundabout way to subtract the number from 10.

Initialize the algorithm. ...

Check to see whether the current value(s) being processed match the base case. ... Redefine the answer in terms of a smaller or simpler sub-problem or

sub-problems.

Run the algorithm on the sub-problem.

Combine the results in the formulation of the answer.

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

Answer: Safe: It can be used without causing harm.

Secure: It can't be hacked.

Reliable: It functions as it should, every time.

Testable: It can be tested at the code level.

Maintainable: It can be maintained, even as your codebase grows.

Portable: It works the same in every environment.

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

Answer: A return is a value that a function returns to the calling script or function when it completes its task. A return value can be any one of the four variable types: handle, integer, object, or string.

Colab paid products - Cancel contracts here