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

2. What is the benefit of lambda?

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

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

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

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

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

**ANSWERS**

1.In Python, an anonymous function means that a function is without a name. As we already know that the *def* keyword is used to define a normal function in Python. Similarly, the *lambda* keyword is used to define an anonymous function in Python. It has the following syntax:

**Syntax:** lambda arguments: expression

* This function can have any number of arguments but only one expression, which is evaluated and returned.
* One is free to use lambda functions wherever function objects are required.
* You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
* It has various uses in particular fields of programming besides other types of expressions in functions.

2.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. When working on **Python** programming you inevitably come across situations where you have to do some data manipulation. In most cases, you use **control statements** to get the desired result, but these control statements can quickly become a bit messy and large. **Functional techniques** can help you write more declarative code that is easier to understand at a glance, refactor, and test. Most of the times it can be much easier to use the **map** , **filter** or **reduce** methods.

The **rule** of thumb you use to determine which method you should use is as follows:

* If you already have a list of values and you want to do the exact same operation on each of the elements in the array and return the same amount of items in the list, in these type of situations it is better to use the map method.
* If you already have list of values but you only want to have items in the array that match certain criteria, in these type of situations it is better to use the filter method.
* If you already have list of values, but you want to use the values in that list to create something completely new, in these type of situations it is better to use the reduce method.

**Map**

Map operation takes a **mapping function** and a vector of data as arguments and returns a new vector, which is the result of applying the mapping function on **each element** of the vector independently. The returned value from map() (map object) then can be passed to functions like **list()** (to create a list), **set()** (to create a set) and so on.

**Syntax**

* function\_to\_apply - map() passes each item of the iterable to this function.
* list\_of\_inputs - iterable which is to be mapped

**example**

**output**

Most of the times map function use lambdas.

**output**

**or**

**output**

**Filter**

The **filter function** operates on a list and returns a subset of that list after applying the filtering rule.

**example**

**output**

**Reduce**

The **reduce function** will transform a given list into a single value by applying a given function continuously to all the elements. It basically keeps operating on pairs of elements until there are no more elements left.

The following example shows how to find the product of given numbers.

**output**

**Using Reduce method:**

**Output**

4.Function annotations introduced in Python 3.0 adds a feature that allows you to add arbitrary metadata to function parameters and return value. Since python 3, function annotations have been officially added to python (PEP-3107). The primary purpose was to have a standard way to link metadata to function parameters and return value.

**Basics of Function Annotations**

Let’s understand some basics of function annotations −

* 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.
* The PEP-3107 makes no attempt to introduce any kind of standard semantics, even for the built-in types. All this work left to the third-party libraries.

5. A recursive function is a [function](https://techterms.com/definition/function) that calls itself during its execution. The process may repeat several times, outputting the result and the end of each [iteration](https://techterms.com/definition/iteration).

The function Count() below uses [recursion](https://techterms.com/definition/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.

function Count (integer N)

if (N <= 0) return "Must be a Positive Integer";

if (N > 9) return "Counting Completed";

else return Count (N+1);

end function

Recursive functions allow programmers to write efficient [programs](https://techterms.com/definition/program) using a minimal amount of code. The downside is that they can cause infinite loops and other unexpected results if not written properly. For example, in the example above, the function is terminated if the number is 0 or less or greater than 9. If proper cases are not included in a recursive function to stop the execution, it will repeat forever, causing the program to crash or become unresponsive.

6. Coding rules and guidelines ensure that software is:

* **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. Call, Indentation, Arguments & Return Values