1.

n python, [def](https://www.geeksforgeeks.org/python-def-keyword/) 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.

# Define function to calculate cube root

# using def keyword

def calculate\_cube\_root(x):

    return x\*\*(1/3)

# Call the def function to calculate cube

# root and print it

print(calculate\_cube\_root(27))

# Define function to check if language is present in

# language list using def keyword

languages = ['Sanskrut', 'English', 'French', 'German']

def check\_language(x):

    if x in languages:

        return True

    return False

# Call the def function to check if keyword 'English'

# is present in the languages list and print it

print(check\_language('English'))

2.

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.

# Define function using lambda for cube root

cube\_root= lambda x: x\*\*(1/3)

# Call the lambda function

print(cube\_root(27))

languages = ['Sanskrut', 'English', 'French', 'German']

# Define function using lambda

l\_check\_language = lambda x: True if x in languages else False

# Call the lambda function

print(l\_check\_language('Sanskrut'))

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.

# Define function using lambda for cube root

cube\_root= lambda x: x\*\*(1/3)

# Call the lambda function

print(cube\_root(27))

languages = ['Sanskrut', 'English', 'French', 'German']

# Define function using lambda

l\_check\_language = lambda x: True if x in languages else False

# Call the lambda function

print(l\_check\_language('Sanskrut'))

2.

Lambda functions **allow you to create small, single-use functions that can save time and space in your code**. They ares also useful when you need to call a function that expects a function as an argument for a callback such as Map() and Filter().

3.

Map takes all objects in a list and allows you to apply a function to it whereas Filter takes all objects in a list and runs that through a function to create a new list with all objects that return True in that function. Hence, knowing the functionality and using it accordingly is important as both the functions have very minute difference in there functionality.

**1). The map() function:**

The map() function is a higher-order function. As previously stated, this function accepts another function and a sequence of ‘iterables’ as parameters and provides output after applying the function to each iterable in the sequence. It has the following syntax:

**SYNTAX:** *map(function, iterables)*

User-defined functions can be sent to the map() method. The user or programmer is the only one who can change the parameters of these functions.

EXAMPLE

def function(a):

return a\*a

x = map(function, (1,2,3,4)) #x is the map object

print(x)

**The filter() function:**

The filter() function is used to generate an output list of values that return true when the function is called. It has the following syntax:

**SYNTAX:** *filter (function, iterables)*

This function like map(), can take user-defined functions and lambda functions as parameters.

EXAMPLE

def func(x):

if x>=3:

return x

y = filter(func, (1,2,3,4))

print(y)

print(list(y))

**Lambda within filter() functions:**

The condition to be checked is defined by the lambda function that is provided as an argument.

EXAMPLE

y = filter(lambda x: (x>=3), (1,2,3,4))

print(list(y))

**The reduce() function:**

The reduce() function applies a provided function to ‘iterables’ and returns a single value, as the name implies.

**SYNTAX:** *reduce(function, iterables)*

The function specifies which expression should be applied to the ‘iterables’ in this case. The function tools module must be used to import this function.

EXAMPLE

from functools import reduce

reduce(lambda a,b: a+b,[23,21,45,98])

4.

**Function Annotations–PEP 3107:**PEP-3107 introduced the concept and syntax for adding arbitrary metadata annotations to Python. It was introduced in Python3 which was previously done using external libraries in python 2.x

**What are Function annotations?**

Function annotations are arbitrary python expressions that are associated with various part of functions. These expressions are evaluated at compile time and have no life in python’s runtime environment. Python does not attach any meaning to these annotations. They take life when interpreted by third party libraries, for example, mypy.

**What are good uses for Python function annotations?**

The type annotations module provides a set of tools for **type checking and type inference of Python code**. It also a provides a set of types useful for annotating functions and objects. These tools are mainly designed to be used by static analyzers such as linters, code completion libraries and IDEs.

5.

A recursive function is **a function in code that refers to itself for execution**. Recursive functions can be simple or elaborate. They allow for more efficient code writing, for instance, in the listing or compiling of sets of numbers, strings or other variables through a single reiterated process.

Recursion is a technique used to solve computer problems by creating a function that calls itself until your program achieves the desired result.

**6. What Are Coding Rules and Guidelines?**

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.**

**Python built-in functions,**

**Python recursion function,**

**Python lambda function, and**

**Python user-defined functions** with their syntax and examples.