**Python Lambda Functions** :

**Python Lambda Functions** are anonymous function means that the 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](https://www.geeksforgeeks.org/python-programming-language/).

**Python Lambda Function 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.

## ****Python Lambda Function Example****

In the example, we defined a lambda function(**upper**) to convert a string to its upper case using [upper()](https://www.geeksforgeeks.org/python-string-upper/).

Input:

str1 **=** 'GeeksforGeeks'

upper **=** **lambda** string: string.upper()

print(upper(str1))

output:

GEEKSFORGEEKS

## ****Module :****

In Python, a module is a single file containing Python definitions and statements. These definitions and statements can include variables, functions, and classes and can be used to organize related functionality into a single, reusable package. Module organizes and reuses code in Python by grouping related code into a single file.Modules can be imported and used in other Python files using the **import** statement.Some popular modules in Python are math, random, csv, and datetime.

**Package:**

Python Packages are collections of modules that provide a set of related functionalities, and these modules are organized in a directory hierarchy. In simple terms, packages in Python are a way of organizing related modules in a single [namespace](https://www.shiksha.com/online-courses/articles/how-to-use-namespaces-in-python/).

* Packages in Python are installed using a package manager like pip (a tool for installing and managing Python packages).
* Each Python package must contain a file named \_init\_.py.
* **Difference Between Module and Package in Python**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Module** | **Package** |
| **Definition** | It can be a simple Python file  (.py extensions) that contains collections of functions and global variables. | A Package is a collection of different  modules with an \_init\_.py file. |
| **Purpose** | Code organization | Code distribution and reuse |
| **Organization** | Code within a single file | Related modules in a directory hierarchy |
| **Sub-modules** | None | Multiple sub-modules and sub-packages |
| **Required Files** | Only Python File  (.py format) | ‘\_init\_.py’ file and python files |
| **How to import** | import **module\_name** | import **package\_name.module\_name** |
| **Example** | math, random, os, datetime, csv | Numpy, Pandas |

## Iterators :

Iterators are objects that can be iterated upon. They serve as a common feature of the Python programming language, neatly tucked away for looping and list comprehensions. Any object that can derive an iterator is known as an iterable.

There is a lot of work that goes into constructing an iterator. For instance, the implementation of each iterator object must consist of an \_\_iter\_\_() and \_\_next\_\_() method. In addition to the prerequisite above, the implementation must also have a way to track the object's internal state and raise a StopIteration exception once no more values can be returned. These rules are known as the **iterator protocol**.

Implementing your own iterator is a drawn-out process, and it is only sometimes necessary. A simpler alternative is to use a generator object. Generators are a special type of function that use the **yield** keyword to return an iterator that may be iterated over, one value at a time.

## Generators :

The most expedient alternative to implementing an iterator is to use a generator. Although generators may look like ordinary [**Python functions**](https://www.datacamp.com/courses/writing-functions-in-python?hl=GB), they are different. For starters, a generator object does not return items. Instead, it uses the **yield** keyword to generate items on the fly. Thus, we can say a generator is a special kind of function that leverages [**lazy evaluation**](https://en.wikipedia.org/wiki/Lazy_evaluation).

Generators do not store their contents in memory as you would expect a typical iterable to do. For example, if the goal were to find all of the factors for a positive integer, we would typically implement a traditional function

# Map:

The basic function of map() is to manipulate iterables. Map executes all the conditions of a function on the items in the iterable. In the above example, we have multiplied each element in the range 0–10 with 2 which gives us completely new list of elements. Map function takes all elements and allows you to apply a function on it and then passes it to the output which can have same as well as different values .

# Filter:

As the name suggests, it is used to filter the iterables as per the conditions. Filter filters the original iterable and passes the items that returns True for the function provided to filter. Therefore only the items in the iterables can be expected to be seen in the output. In the above example the condition is given in the form of lambda function and the elements which satisfy the condition are given in the list. The elements which are divisible by 2 are left and others are filtered out.

