ASSIGNMENT-1

**Name:**A.B.keerthana.

**Reg.no.:**121011012743

**Subject:**datascience.

**Date:**30.08.2023

**1.LAMBDA FUNCTIONS**

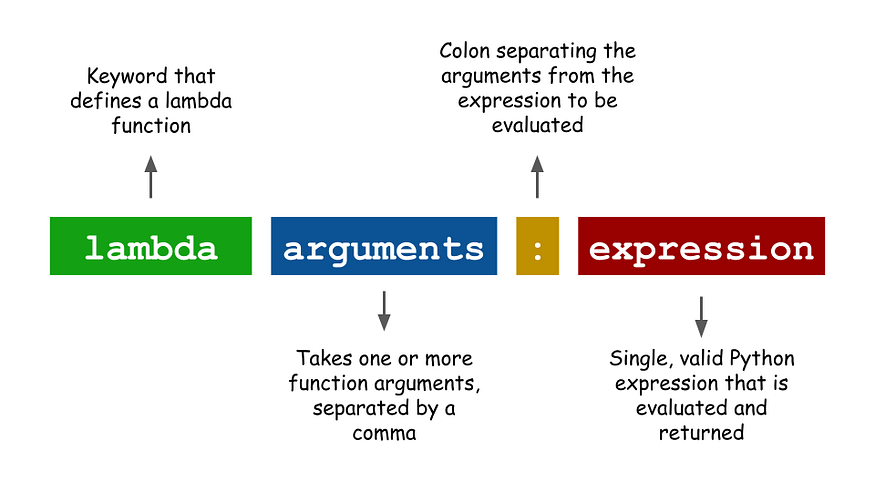
Lambda functions in Python are small, **anonymous** (or nameless) functions characterized by a more concise syntax compared to regular Python functions.

Within the realm of data science, lambda functions are frequently used in conjunction with **higher-order functions**. These are functions that take one or more functions as arguments, or return a function as their result. Common examples include map(), filter(), and reduce().

Before diving into the applications, let’s first take a look at the syntax of lambda functions.

**Syntax**

In order to use a lambda function in Python, the following **four components** are required:



* **lambda:** The keyword that defines a lambda function.
* **Arguments:** Lambda functions can take one or more arguments. If the number of arguments is more than one, they need to be separated by a comma.
* **Colon:** This serves as a separator between the function arguments and the expression.
* **Expression:**A single and valid Python expressionwhich is going to be evaluated and returned as the result of the function.
* Applying this syntax, we can produce a simple example of a lambda function that takes two input arguments, xand y, and returns their sum:
* >>> (lambda x, y: x + y)(3, 5)  
  8
* By using parentheses around the lambda function, we can make use of a principle called [immediately invoked function expression](https://en.wikipedia.org/wiki/Immediately_invoked_function_expression), which allows us to define and immediately execute a function.
* Since a lambda function is an expression, we can also assign a name to it. Below is an example of the same function as above, but this time we call it func.
* >>> func = lambda x, y: x+y  
  >>> func(3, 5)  
  8

**2.DIFFERENCE BETWEEN PACKAGES AND MODULES.**

## What is the Difference between Module and Package in Python**?**

The main difference between a module and a package in Python is that a module is a simple Python script with a **.py** extension file that contains collections of functions and global variables. In contrast, a package is a directory that contains a collection of modules, and this directory also contains an \_\_init\_\_.py file by which the interpreter interprets it as a package.

## What is a Module in Python?

[In Python](https://www.scaler.com/topics/python/), a module can be a simple python file (**.py** extension file), i.e., a combination of numerous functions that we can use to provide different functionalities in a program.

Python Modules are essentially Python Programming Statements containing various types of Python Functions used to perform various operations in a Python Program. In the script, Python modules serve as a ready-made library available to programmers and users.

**Let’s see an example to understand Python Modules:**

Make a new file with any name, say it is *scalerAcad*.*py*, and save the code below in this file.

**scalerAcad.py**

def module\_function(para):

print("Creating a new Module: " + para)

Make a new file with any name, say *test*.*py*, and paste the code below into it. Run it.

**test.py**

import scaler\_acad

scaler\_acad.module\_function("Scaler")

**Output:**

Creating a new Module: Scaler

**Explanation:**  
In the above code, we create a new module named *scalerAcad*.*py* by creating a new Python file consisting of a function named **module\_function**, which takes a parameter and prints a statement.

Therefore, after creating the module, we use it in the *test*.*py* file by importing it, calling it with the parameters, and printing the statement.

## What is a Package in Python?

As we discussed earlier, to create large-scale-based real-world applications, we divide large code into smaller pieces to perform different functionalities, resulting in many modules. To collaborate with all of the modules, we create a Python package with an \_\_init\_\_.py file that informs the Python Interpreter that the given folder is a Python Package.

For any source code, a Python package serves as a user-variable interface. This functionality enables any functional runtime script to use a Python package at a specified moment, showing the main difference between the module and the package in Python.

**To import a package, we type the following:**

import math

In the above code, **math** is a package.

Only its immediate modules are imported when we import a package, not the sub-packages. If you try to access those, it will raise an AttributeError.

## What Makes Python Package Different from Modules?

A Python package defines the code as a separate unit for each function when using a library. While the modules themselves are a distinct library with built-in functionality, the advantage of packages over modules is their **reusability**. So this is the difference between a module and a package in Python.

### Explicit Namespaces

It gives the program, which is interpreted for the first time, the default namespace. These namespaces serve as the source code for the coding's identification. However, a novice programmer can also integrate them from the library. However, it is always recommended to be familiar with general namespaces to execute code correctly.

**Code:**

def Acad():

para = "Scaler"

Acad()

**Output:**

>>> Acad

<function namespace at 0x000001F91DE9FF70>

>>>

### Convenience API

Generally, this is a way to namespace specific code objects. It takes the user right to the core of the code, making it simple to see problems as well. Additionally, it aids in interpreting the codes to be used as user interface codes when needed.

**Code:**

import hello

hello.hey()

**Output:**

Error

**3.WHETHER DIVISION METHOD IS AVAILABLE IN MATRICES?**

### Iterator

An iterator is an object which contains a countable number of values and it is used to iterate over iterable objects like list, tuples, sets, etc. Iterators are implemented using a class and a local variable for iterating is not required here, It follows lazy evaluation where the evaluation of the expression will be on hold and stored in the memory until the item is called specifically which helps us to avoid repeated evaluation. As lazy evaluation is implemented, it requires only 1 memory location to process the value and when we are using a large dataset then, wastage of RAM space will be reduced the need to load the entire dataset at the same time will not be there.

Using an iterator-

* iter() keyword is used to create an iterator containing an iterable object.
* next() keyword is used to call the next element in the iterable object.
* After the iterable object is completed, to use them again reassign them to the same object.

### **Generators**

It is another way of creating iterators in a simple way where it uses the keyword “yield” instead of returning it in a defined function. Generators are implemented using a function. Just as iterators, generators also follow lazy evaluation. Here, the yield function returns the data without affecting or exiting the function. It will return a sequence of data in an iterable format where we need to iterate over the sequence to use the data as they won’t store the entire sequence in the memory.

## Table of difference between Iterator vs Generators

| Iterator | Generator |
| --- | --- |
| Class is used to implement an iterator | Function is used to implement a generator. |
| Local Variables aren’t used here. | All the local variables before the yield function are stored. |
| Iterators are used mostly to iterate or convert other objects to an iterator using iter() function. | Generators are mostly used in loops to generate an iterator by returning all the values in the loop without affecting the iteration of the loop |
| Iterator uses iter() and next() functions | Generator uses yield keyword |
| Every iterator is not a generator | Every generator is an iterator |

**5.map() and filter() functions**.

**map()** function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

**Syntax :**

map(fun, iter)

**Parameters :**

***fun :****It is a function to which map passes each element of given iterable.****iter :****It is a iterable which is to be mapped.*

**NOTE :** You can pass one or more iterable to the map() function.

The **filter()** method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

## **Python filter() Syntax**

The filter() method in [Python](https://www.geeksforgeeks.org/python-programming-language/) has the following syntax:

***Syntax:****filter(function, sequence)*

***Parameters:***

* ***function:****function that tests if each element of a sequence is true or not.*
* ***sequence:****sequence which needs to be filtered, it can be sets, lists, tuples, or containers of any iterators.*

***Returns:****an iterator that is already filtered.*