

NAME : M.MANOJ

REG NO : 121012012747

SUBJECT : APPLIED AI

SUBJECT CODE : XCSHDO3

DEPARTMENT : B.TECH(CSE)

SEMESTER : 5TH SEM

**LAMBDA FUNCTIONS:**

Lambda functions are similar to user-defined functions but without a name. They're commonly referred to as anonymous functions.

Lambda functions are efficient whenever you want to create a function that will only contain simple expressions – that is, expressions that are usually a single line of a statement.

Every anonymous function you define in Python will have 3 essential parts:

* The lambda keyword.
* The parameters (or bound variables), and
* The function body.

A lambda function can have any number of parameters, but the function body can only contain one expression. Moreover, a lambda is written in a single line of code and can also be invoked immediately.

**Syntax :**

The formal syntax to write a lambda function is as given below:

lambda p1, p2: expression

**example :**

adder = lambda x, y: x + y

print (adder (1, 2))

**The output for the above example** is 3.

### **Code Explanation for above example :**

Here, we define a variable that will hold the result returned by the lambda function.

**1.** The lambda keyword used to define an anonymous function.

**2.** x and y are the parameters that we pass to the lambda function.

**3.** This is the body of the function, which adds the 2 parameters we passed. Notice that it is a single expression. You cannot write multiple statements in the body of a lambda function.

**4.** We call the function and print the returned value.

**DEF PACKAGES AND MODULES:**

In Python, both modules and packages organize and structure the code but serve different purposes.

In simple terms, a module is a single file containing python code, whereas a package is a collection of modules that are organized in a directory hierarchy.

**modules :**

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.

**Example:**

Consider a Python module math.py that contains a function to calculate the square of a number.

#math.py module

def square(i):

return x\*\*2

This module can be used be imported and used in the different files as follows:

#main.py file

import math

print(math.square(5))

**The output for the above code** is 25.

**Packages :**

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.

* 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.

**Example :**

Let there be any package (named my\_package) that contains two sub-modules (mod\_1, and mod\_2)

my\_package/

\_init\_.py

mod\_1.py

mod\_2.py

**Note:**

 init.py file is required to make python treat the dictionary as a package.

**MATRIX DIVISION:**

For matrices, there is no such thing as division. You can add, subtract, and multiply matrices, but you cannot divide them. There is a related concept, though, which is called “inversion”.but we can perform division operation in array.

Python's numpy. divide() computes the element-wise division of array elements. The elements in the first array are divided by the elements in the second array.

**SYNTAX :**

numpy.divide(arr1, arr2, out = None, where = True, casting = ‘same\_kind’, order = ‘K’, dtype = None) :

Array element from first array is divided by elements from second element (all happens element-wise). Both arr1 and arr2 must have same shape and element in arr2 must not be zero; otherwise it will raise an error.

**EXAMPLE 1 FOR (arr1 divide by arr2 elements):**

# Python program explaining

# divide() function

import numpy as np

# input\_array

arr1 = [2, 27, 2, 21, 23]

arr2 = [2, 3, 4, 5, 6]

print ("arr1 : ", arr1)

print ("arr2 : ", arr2)

# output\_array

out = np.divide(arr1, arr2)

print ("\nOutput array : \n", out)

**OUTPUT FOR THE EXAMPLE 1:**

arr1 : [2, 27, 2, 21, 23]

arr2 : [2, 3, 4, 5, 6]

Output array :

[ 1. 9. 0.5 4.2 3.83333333]

**EXAMPLE 2 FOR (elements of arr1 divided by divisor):**

# Python program explaining

# divide() function

import numpy as np

# input\_array

arr1 = [2, 27, 2, 21, 23]

divisor = 3

print ("arr1 : ", arr1)

# output\_array

out = np.divide(arr1, divisor)

print ("\nOutput array : \n", out)

OUTPUT FOR THE ABOVE EXAMPLE 2:

arr1 : [2, 27, 2, 21, 23]

Output array :

[ 0.66666667 9. 0.66666667 7. 7.66666667]