**Python functions Contd.**

Please see the code

**def main():**

**print(“Welcome to the world of python”)**

**print(“Python Guru”)**

In this program there are two print statements. When you execute this

code you will get only Python Guru printed. Main function is the entry point of any program. But python interpreter executes the source file code sequentially and doesn’t call any method if it is not part of the code. But if it is directly part of the code then it will be executed when the file is imported as a module.

That’s why there is a special technique to define main method in

python program, so that it gets executed only when the program is run

directly and not executed when imported as a module.

Let’s see how to define python main function in a simple program.

It is important that after defining the main function, you call the code

by

**if\_\_name\_\_== "\_\_main\_\_"**

and then run the code, only then you will get the desired output.

**def main():**

**print(“Welcome to the world of python”)**

**print(“Python Guru”)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

The **output** of the above program will be as expected.

**Python Guru**

**Welcome to the world of python**

* When a python program is executed, python interpreter starts

executing code inside it. It also sets few implicit variable values,

one of them is \_\_name\_\_ whose value is set as \_\_main\_\_.

* For python main function, we have to define a function and then

use if \_\_name\_\_ == '\_\_main\_\_'condition to execute this function.

* If the python source file is imported as module, python interpreter

sets the \_\_name\_\_ value to module name, so the if condition will

return false and main method will not be executed.

* Python provides us flexibility to keep any name for main method,

however it"s best practice to name it as main() method.

**Anonymous functions**

Python supports functions with out name. These functions are called

anonymous functions. Anonymous functions are created using lambda

expression in python

For example if want to create a function that adds two numbers and

returns the sum.

**def addnum(a,b):**

**return a+b**

Here we defined a new function named addnum. We can use this

function as follows:

**f = addnum(10,20)**

**print(f)**

The above function is not an anonymous function. It has a specific

name, addnum.

We can create the same function using lambda expression in python.

**lambda a,b:a+b**

There is no specific name for this function. Then how to call this

function?

For this purpose we can assign this function to sum variable.

For example

**x = lambda a,b:a+b**

Now we can use this function as follows

**x(10,20)**

**We will get the value 30.**

Another example for using lambda expression inside a function

Solution for quadratic equation

**def qe(a,b,c):**

**return lambda x:a\*x\*\*2+b\*x+c**

**If you want to calculate the quadratic equation**

**2x 2 +3x-5 where x=0,1,2**

**f = qe(2,3,-5)**

**print f(0)**

**print f(1)**

**print f(2)**

**or**

**print qe(2,3,-5)(0)**

**print qe(2,3,-5)(1)**

**print qe(2,3,-5)(2)**

**Why do we need lambda?**

Actually, we don’t absolutely need lambda; we could get along without it. But

there are certain situations where it makes writing code a bit easier, and the

written code a bit cleaner.

**What kind of situations?**

Situations in which

(a) the function is fairly simple, and

(b) it is going to be used only once.

Normally, functions are created for one of two purposes:

(a) to reduce code duplication, or

(b) to modularise code.

* If your application contains duplicate chunks of code in various places, then you can put one copy of that code into a function, give the function a name, and then using that function name call it from various places in your code.
* If you have a chunk of code that performs one well-defined operation but is really long and gnarly and interrupts the otherwise readable flow of your program then you can pull that long gnarly code out and put it into a function all by itself.
* But suppose you need to create a function that is going to be used only once called from only one place in your application. Well, first of all, you don"t need to give the function a name. It can be “anonymous”. And you can just define it right in the place where you want to use it. That"s where lambda is useful.
* First of all- Why would you want a function that is called only once? That

eliminates reason (a) for making a function.

* Then- the body of a lambda can contain only a single expression. That means that lambdas must be short. So that eliminates reason (b) for making a function.

**Write a single line program to sort the following list using lambda**

**expression.**

**List = ["Issac Newton","Issac Asimov","Orson Scott Card","Brain**

**Mercurry","Vladimir Illich Lenin","Arthus C. Clarke"]**

**sort it by last name.**

**The output must be as follows**

**['Issac Asimov', 'Orson Scott Card', 'Arthus C. Clarke',**

**'Vladimir Illich Lenin', 'Brain Mercurry', 'Issac Newton']**

**map Function**

The map() function applies a given function to each item of an iterable (list, tuple etc.) and returns a list of the results.

**Syntax**

**map(function,iterable,....)**

The map() function applies a given function to each item of an iterable and returns a list of the results.

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.

**For example**

**def addnum(a,b):**

**return a+b**

**l1=[1,2,3,4,5,6,7]**

**l2=[2,3,4,5,6,7,9]**

**x=map(addnum,l1,l2)**

**print(x)**

**Output**

**[3, 5, 7, 9, 11, 13, 16]**

**Using lambda expression**

**l1=[1,2,3,4,5,6,7]**

**l2=[2,3,4,5,6,7,9]**

**x = map(lambda a,b:a+b,l1,l2)**

**print(x)**

**filter function**

If you want to select some specific numbers from a list of numbers you

can use the filter function.

**For example**

l1=[1,-2,3,-4,5,6,-7]

**to filter negative numbers**

**x = filter(lambda x:x<0,l1)**

**print(x)**

**the output will be**

**[-2, -4, -7]**

If you want to **eliminate zeros** from a list of numbers, we can use the

**filter function as follows**

**n = [1,2,0,3,0,5,2,0,4]**

**x = filter(None,n)**

**print(x)**

**output**

**[1,2,3,5,2,4]**

**If you want to filter missing data from a list of data, use filter as follows**

**name = [“”,”Hadi”,””,”Saji”,”Eza”,””,””,”Lajish”]**

**n = filter(None,name)**

**print(n)**

**Output**

**[“Hadi”,”Saji”,”Eza”,”Lajish”]**

**reduce function**

Suppose we have a sequence of data. And we want to apply a function

on it’s first two values and then apply the same function to the result of

the previous action and the next value of the sequence. We can create

loops and do t this problem. I will show you how to solve this problem

using reduce function.

**L = [1,10,20,30,12,3,2,9]**

**if we want to add these numbers as follows**

**1+10 =11**

**11 + 20 = 31**

**31 + 30 = 61**

**61 + 12 = 73**

**73 + 3 = 76**

**76 + 2 = 78**

**78 + 9 = 87**

**use reduce function as follows**

**n=reduce(lambda x,y:x+y,L)**

**print(n)**

**Output**

**87**

**You will get the same result using for loop**

**s=0**

**for x in L:**

**s = s+x**

**print(s)**

**Command Line arguments in python**

Python command line arguments are input parameters passed to the

script while executing as in the case of other programming languages.

There are many options to read python command line arguments.

Some of them are listed below.

**1. Python sys.argv**

**import sys**

**for i in sys.argv:**

**print(i)**

**sys.arv[0] is file name itself and sys.argv[1] will be the 1 st parameter and so on.**

**For example**

if we execute this program by giving

**python test.py kerala calicut**

**output will be**

**test.py**

**kerala**

**calicut**

**here sys.argv[0] is file name and kerala and calicut are next two arguments.**

**Two other options are**

**1. using getopt module and argparse module**

**Expalin these two modules with examples.**

**JSON format in python**

JSON(Java script object notation) format is widely used now a days for data communication. It’s small and light weight data format. It is almost like python dictionary. It is an ideal data format for transporting data between server and client. Sample format is given below.

**{**

**“title” : “Oru deshathinte kadha”,**

**“release\_year” : 1990’,**

**“author” : “S K Pottakkad”,**

**“pages” : 500,**

**“co\_auther” : Null,**

**“out\_of\_stock” : false,**

**“credits” : {**

**“Publisher” : “NBS”,**

**“Illustration” : “Namboothiri”,**

**}**

**}**

json package is used to manipulate the json format in python.

Load method of jason package is used to read a json formatted data

**from a file.**

**Import json**

**jf = open(“file\_name”,”r”,encoding=”utf-8”)**

**book= json.load(jf)**

**jf.close()**

**Here the json formatted data will be stored in book as a python dictionary.**

If the same data will be in a **string** format we can load this data using

**loads method**.

If you want to convert this python **dictionary** back to json format use

**dump or dumps method**.

**Study csv and json modules completely and write a short note about all its methods with examples.**