PHY 2C 08 :COMPUTATIONAL PHYSICS

(Prepared for II Sem. M.Sc(Physics)-CUCSS Syllabus -w.e.from 2012 admn.

P.A.Sivaramakrishnan
Associate Prof.of Physics
Govt.Arts & Science College
Kozhikode-673 018

pa.sivaramakrishnan@gmail.com

MODULE-1
INTRODUCTION TO PYTHON LANGUAGE

Introduction:

Python is a simple high level language with clean sytax. It has extensive standard libraries and can be learned in a few days.

Python Variable:

A variable is a name that represents any data whose value may change during the execution of the program. Any single character/ name(string)/(string+numeric) can be given to a variable except some reserved key words which are used for specific pruposes in Python like 'while' ,'if','else',etc. It is better to use meaningful names as variables.
Ex: sum, sum_value, area, area12, vol,VOL,.........
Caution: vol and VOL are two different variables

Data types and variables in Python:

Python supports numeric data types like integers,floating point numbers and complex numbers. To handle character strings, it uses string data types. It also supports other data types like lists, tuples & dictionaries. So, the varies data types in python are integers, floating point numbers , strings,complex numbers, lists,tuples, dictionaries(compound data types).

In languages like C,C++ and Java, we need to explicitly declare the type of a variable. This is not required in Python. The data type of a varible is decided by the value assigned to it. This is called dynamic data typing. The type of a particular variable can change during the execution of the program. If required one type of variable can be converted into another type by explicit type casting, like y='1234' ,y is a string data type
z=float(y), etc , here y is converted to float data type

(Strings are enclosed within single quotes or double quotes)

INTERPRETER MODE

(to get the interpreter mode , go to terminal and type 'python' and press enter key, you will get a prompt >>> like this. Now you are in the interpreter mode. To get out from this mode, press crtl+D)

```
>>x=10
>>print x, type(x)   #print x and its type
>>x=10.5
>>print x ,type(x)
>>x=3+4j
>>print x,type(x)

>>x='I am a string'
>>print x,type(x)
```

The output of the above python statements are given below. Note that the type of the variable x changes during the execusion of the program , depending on the value assigned to it.

```
10  <type 'int'>
10.5 <type' float'>
(3+4j)  <type 'complex'>
I am a  string' <type 'str'>
```

Expression statement in Python:

An expression statement is one that has the following form , like x=10,y=x+2,z=x+y**2,etc

Operators and their Precedence

Python supports a large no.of arithmetic, logical and relational operators.

(a) Arimetic operators(in hierarchy) :

Ist	**	exponentation
II nd	* , / , %	Multiplication , division, Modulo division
III rd	+, -	addition ,subtraction

explanation:

In the expression 2+3*4, the multiplication is performed first then addition. So the output will be

14. If we want the addition to be done first, enforce it by using parenthesis like (2+3)*4. So the output will be 24. When ever there is ambiguity in evaluation, use parenthesis to clarify the order of evaluation.

(b) Relational operators:

less than	<
less than or equal to	<=
greater than	>
greater than or equal to	>=
equal to	= =
not equal to	!= or < >

(c)Uniary operators:

x+=1 means x=x+1, here previous value of x is incremented by 1 -(increment operator)

x-=1 means x=x-1 ,here previous value of x is decremented by 1 -(decrement operator)

x*= 2 means x=x*2, here previous value of x is multiplied by 2- (multiplication operator)

x/=2 means x=x/2 ,here previous value of x is divided by 2- (division operator)

(d) Boolian or Logical operators:

OR	or
AND	and
NOT	not

Python strings:

The string data type is somewhat different from other data types. It is a collection of same kind of elements, characters. The individual elements of a string can be accessed by indexing as shown. String is a compound or collection data type.

Ex:1

```
>>s='hello world'
>>print s[0]
>>print s[1]
>>print s[-1]
```

The out put will be as shown below:

```
h
l
d
```

Ex:2

```
a='hello'+' world'
print a
b='ha'*3
print b
print a[-1]+b[0]
```

The output will be as shown below:

```
helloworld
hahaha
dh
```

PYTHON AS A CALCULATOR(USE INTERPRETER MODE)

```
>>10+2
>>12-2
>>13*3

>>15/3
>>2**3
>>3%2
......
......
```

Slicing of string:

Part of a string can be extracted using the slicing operation. It can be considered as a modified form of indexing a single character. Indexing using s[a,b] extracts elements s[a] to s[b-1] . We can skip one of the indices. If the index on the left side of the colon is skipped, slicing starts from the first element and if the index on the right side is skipped, slicing ends with the last element.

Ex:1

```
a='hello world'
print a[3:5]
print a[6:]
print a[:5]
```

The outputs will be as shown below:

```
lo
world
hello
```

<u>LISTS IN PYTHON:</u>

List is an important data type in python and much more flexible than strings. The elements of a list can be any data type like integer,float,complex, string or even another list. Lists are defined by enclosing the elements inside a pair of square brackets, separated by commas. A list can be defined in a program just like any other variable .For eg:

a=[1,2,3] defines a list with three integer elements in it.

B=[1,2,33,3+2j,'hello world'] is a mixed list with 5 elements
c=[ ] is an empty list with 0 elements in it.

List addition and multiplication are demonstrated by the following example. We can also have another list as an element of a list. Also, List can be sliced in a manner similar to that of strings

Eg:

```
 a=[1,2]
print a*2
print a+[3,4]
b=[10,20,a]
```

```
print b
```

The output will be as shown below:

```
[1,2,1,2]
[1,2,3,4]
[10,20,[1,2]]
```

Mutable and immutable types:

There is one major difference between string and list types. List is mutable but string is not. We can change the value of an element in a list, add new elements to it and remove any existing element. This is not possible with string type.

Eg:

```
s=[3,3.5,234]
s[2]='hello'
print s
x='my name'
x[1]=2
```

verify out put ?

FUNCTIONS AND MODULES:

A function is generally an isolated group of statements(codes) that has a name and has a specific job.

Functions two types:

1. Built- in function
2. User definded functions

BUILT-IN FUNCTIONS:

Built-in functions are those which are already written and kept in the libaray of Python and can be readly used. .Built-in function are again two types.

1.General functions **2.Mathematical functions.**

Examples of General functions:

(a) len( ) means the length(total elements) in a list.

Eg:1 a=[1,2,5]
len(a)

eg:2
len('grammer')

(b) append( ) means to add a new element to the position after the last element of a list.

Eg: a=[1,2,5]
a.append(6)

(c) insert( ) means to insert a new element at the desired position in a list.

Eg:1
```
 a=[1,2,5]
a.insert(0,9)
print a
```

eg:2
```
 a=[1,2,5]
a.insert(3,8)
```

(d) del (name of the list) means to delete a list completely

eg:
```
 a=[1,2,5]
print a

del a
```

(e) remove( ) means to remove an existing element from the list(if a particular element is duplicating, this function will remove one at a time)

eg:1
```
a=[1,2,3,4,5]
a.remove(1)
print a
```

eg:2
```
 a=[1,2,4,45,6]
a.remove(4)
print a
```

(f) reverse( ) means to reverse a list completely

eg:1
```
a=[1,2,3,4,5]

a.reverse( )
print a
```

eg:2

```
b=[1,1,[1,2],3,5]
b.reverse( )
print b
```

(g) sort( ) menas to sort a list in the ascending order

eg;

```
  a=[4,6,3,7,5,9]
a.sort()
```

(h) max() means to pick the maximum of members in a list

eg: a=[3,4,6,87,99]
max(a)

(i) min() means to pick the minimum of members in list
eg: a=[3,4,6,87,99]
min(a)
(i) sum( ) means to find the sum of the members in a list

```
a=[1,2,3,4,5]
sum(a)
```

(j) count() means to count the occurrence of an element in a list

```
a=[1,2,3,3,4,5]

a.count(3)
```

(k) int(3.5) means to find the interger part

(l) float(3) means to convert int into float

(m)pop() means remove the last element of a list and show it on screen

eg: a=[1,2,3,4]
find
a.pop( )
a.pop(0)
a.pop(2)

(n) extend() means to extend a given list with another

```
a=[1,2,3]
b=[4,5,6]
a.extend(b)
```

(o) range():
The general syntax is

range(start index, stop index, increment/decrement index): range function generates a set of number number from (start index to stopindex-1) with increment/decrement specified.
Ex;
range(3,10,1); generates nos from 3 to 9 with increment 1. ie,it generates nos. 3,4,5,6,7,8,9
If there is only one argument in the range function, then it will be assumed that the start index is 0 and increment is 1
eg:
range(10) generates nos. 0,1,2,3,4,5,6,7,8,9

do more examples with range function.

(p) round(f,n): means it will convert a floating point no. 'f' truncated to 'n' decimal places

<u>in and not in :</u> in and not in are test statements (operators) to find whether a particular element is in the list or not. The operator will return true or false depending on the test result.

Eg: a=[1,2,3,4,5]
5 in a

eg:

12 in a

eg:

13 not in a

<u>Examples of mathematical functions:</u>

Pow(2,4)= 2^4 means raising power

hypot(3,4)=5 means squart root of (3^2+4^2)

sin( ),cos( ) ,tan( ),asin(),acos(),atan(),ceil(),log(),log10( ), exp( ), sqrt()

floor(),ceil(),factorial( ),radians(),degrees(),abs(),fabs(),complex(2,3),abs(complex(2,3),.........................

<u>2,USER DEFINED FUNCTIONS:</u>

User can define own function according to the requirement. Such a function can be defined using the combination 'def ' and ' return' statements. The statement 'def' is followed by the form of the function and a colon:The indented block of code creates the function. The return statement decides what value is passed to the program when the functon is called.

```
def  f(x):
          return (x**3+2*x+2)
```

print f(2)

SET FUNCTION:

A set function can be constructed by defining the elements of the set in the following way.

s=set([4,6,7,8])

(a) add()

An element can be added to the set by**add( )** function

s.add(2)

so the output will be set([2,4,6,7,8])

again, s.add(4),

we can see that the output remains the same as before since add function will have no effect on duplicate element.

(b) update()

The same behaviour occures in the 'update'function..

eg:

s.update([2])

so the output will be set([2,4,6,7,8])

(c): **pop() :** means to remove the first element from a given set

eg: s=set([1,2,3,4,5])

s.pop()

the output will be 1

print s

set([2,3,4,5])

(**d**)**remove()** means to remove a specified element from the given set

eg: s=set([5,6,7,8,9])

s.remove(9)

the output will be set([5,6,7,8])

clear() : means to remove all elements from a set

eg:s.clear()

print s([])

the out put will be

(e) in and not in

we can check if an object is in the set using the **in and not in** operator

eg:

s=set([1,2,3])

2 in s

true

45 not in s

true

(f)) Union

The union is the merger of two sets. Any element in sets s1 or s2 will appear in their union

eg; s1=set([1,2,3])

    s2=set([2,4,5])

s1.union(s2) or s1|s2

o/p

set([1,2,3,4,5])

(g) intersection

Any element which is in both the sets s1 and s2 will appear in their intersection

eg; s1=set([1,2,3])

    s2=set([2,4,5])

s1.intersection(s2) or s1 & s

o/p

set([2])

(h) symmetric difference

The symmetric difference of two sets is the set of elements which are in one of either set, but not in both.

Eg:

s1=set([4,6,9])

s2=set([1,6,8])

s1.symmetric_difference(s2) or s1^s2

o/p

set(8,1,4,9])

TUPLES

A tuple is just like a list,that can hold an arbitary number of objects of different data types. However, tuples are immutable, i.e, their elements cannot be modified and elements cannot be added to or removed from the tuple. However, an individual element of a tuple can be accessed. Tuples can be created by specifing a set of objects separted by commas and enclosed in ordinary parenthesis

eg:

a=1,2,3,'hello'
print a,type(a)
The o/p will be
(1,2,3,'hello')   <type 'tuple'>

eg:
b=([,2,3,'hello')
print b, type(b)
To access a particular element of tuple use the code, b[2] to get the element 'hello'
<u>question</u>
Can you try adding or removing an element in a tuple and see yourself how an error message is generated?

<u>Dictionaries</u>

Dictionary is a special kind of list with paired elements. Every element has two parts separated by ':'sign. Dictionaries can be created by specifying a set of paired objects separated by commas in a pair of braces {}. It is a group of {key:value} pairs. The elements in a dictionary are indexed by keys. Keys in a dictionary are required to be unique. Keys can be almost any Python type,but are usually numbers or strings. Values ,on the other hand , can be any arbitary Python object.

This is convenient for saving quantities like phone book entries, student scores, etc.
phonebook={}   creates an empty dictionary named phonebook
phonebook['Raj']= 9446459234 .here Raj is the key and 9446459234 is the value.
You can check the contents in the dictionary name 'phonebook' by typing 'phonebook'
Again, phonebook['Ram']= 9945345452, you can check the contents of the dictionary now by typing 'phonebook'. Like you can add many keys and values as possible. Now , check the different keys in the dictionary by typing 'phonebook.keys(). Check the output. Check the different values in the dictionary by typing phonebook.values(). Check the output. Try to find the value corresponding to key named 'Raj'
phonebook={'Raj':9446459234,'Ram':9945345452,'mini';8890443489,'earth':'heaven','a':'apple'}

<u>Input from keyboard</u>

Most of the programs require some input from the user. The are mainly two functions used for this purpose, *input()* for numeric type data and *raw_input()* for string type data. A message to be displayed can be given as an argument while calling these functions.
Eg:

```
x=input('enter an integer')
y=input('enter one more ...')
print 'the sum is',x+y
s=raw_input('enter a string')
print 'you entered ',s
```

It is possible to read more than one variable using a single input() function. String type data is read using raw_input() may be converted into integer or float type data if contain only valid characters.

Eg:

```
x,y==input('enter x and y separated by commas')
print 'The sum is',x+y
s=raw_input('enter a decimal no.')
a=float(s)  #converts string into float  data type
print s*2   #prints string twice
print a*2   #print value times  2
```

Conditional Execution:

The most fundamental aspect of a programming language is the ability to control the sequence of operations. One of this control is the ability to select one action form a set of specified alternatives. The other one is the facility to repeat a series of actions any number of times or till some condition becomes false. To execute some section of the program code only,if certain conditions are true, Python uses *if,elif,......else*,construct.

```
Eg: >>> x=input('enter a number')
>>> x=input('enter a number')
>>> if x>10:
        print   x,' is greater than 10"   # note the colon and indentation
 >>>elif x<10:
        print  x,' is smaller than 10'
>>>else:
         print ,x' is equal to 10'
```

eg:2

To find the largest of two nos.

```
x,y=input('enter two numbers separated by commas')
if x>y:
    print x ,'is the biggest no.'
else:
    print y,'is the biggest no.'
```

eg:3

To find the largest of three nos.

```
x,y,z=input('enter three nos.separated by commas')
if x>y:
     if  x>z :
          print x,'is the   biggest no.
     else:
          print z,'is the biggest no.'
else:
     if y>z:
           print y,'is the biggest no.'
     else:
           print z,'is the biggest no.'
```

ALTERNATE PROGRAM:

```
x,y,z=input('enter three nos.separated by commas')
if x<y:
   x,y=y,x
if x<z:
   x,z=z,x
           print x,'is the biggest no.'
```

do as my examples as possible.

ITERATIONS AND LOOPING

When a condition remains true,if a set of statements are to be repeated, the *while* and *for* constructs are employed. The general syntax of a *while* and *for* loop may given as follows:

```
set while condition:                        |       for  elements  in list or tuple :
         set of statements to be repeated    |              set of statements to be  repeated
            (body of the loop)                                  (body of the loop)
```

Note the indentation of the statements to be repeated

(a) Iteration with while loop

eg: 1

```
x=10
while x>0:
        print x,
        x-=1
print'while loop terminated'
```

It is possible to have one while loop(inner loop) within body of the another while loop(outer loop) and so on.......called nested loop.

Eg:2 Multiplication table

```
x=10
while x>0:
    y=10
    while y>0:
        print x,'*',y,'=',x*y
        y-=1
    print'-------------------------------------'
    x-=1
```

eg:3: sum of numbers

```
sum=0
x=0
while x<=10:
    sum+=x
    x+=1
print sum
```

do more examples with while loop statements

(b)Iteration with for loop (using range function)

eg:1

```
for i in range(10,0,-1):
        print i
```

eg : 2 Multiplication table

```
for i in range(1,11):
    for j in range(1,11):
        print i,'*',j,'=',i*j
```

```
  print '----------------------'
```

eg;3To sort a set of numbers:

```
a=[ ]
n=input('give the count of numbers:    ')
for i in range(n):
     a.append(input('type the numbers:    '))
a.sort()
print a
a.reverse()
print a
print max(a)
print min(a)
print sum(a)
```

eg:3 to find the sum of numbers

```
sum=0
for i in range(0,11):
   sum+=i
print sum
```

<u>**If.....break condition**</u>

If ...break condition statement is used to terminate a loop, if some condition is met.

Eg:1

```
x=100
i=0
while i<=100:
    x/=2
    i+=1
    print x
```

verify the o/p

In the above eg; if we don't want the zeros, we can modify the above program using if.... break statement.

Eg:2

```
x=100
```

```
i=0
while i<=100:
   x/=2
   i+=1
   if x<>0:
       print x
   else:
       break
```

Note: the break statement will exit from the inner loop

eg:3

```
for i in range(0,3):
  for j in range(0,3):
       if (i= =j):
           break
       else:
           print i,'*',j,'=',i*j
```

<u>if.....continue statement</u>

if...continue statement is used to skip the rest of the statement in a loop and to go the beginning of the loop, if a particular is true.

```
x=0
i=0
while i<100:
  i+=1
  x+=1
  if x>25:
  print' skipping work',i
   continue
  print x
 print 'loop exit after ' ,i,'th iteration'
```

do more examples

MODULES:

In Python, the definitions of functions may be saved in a file and use them in a script mode or in an interpreter just as header files in C-language. Such a file is called a ***module*** . Generallly, the module name is appended with function name. Definitions from a module can be imported into other modules or into the main module. Examples of some standard modules are math,random,pylab,numpy,scipy,etc. The main advantage of creating and using modules is that longer programs can be split into several files so that maintenance of code is easy and can be reused in several programs by including the module with keyword *import* at the beginning of the program.

Different ways of *importing modules*

1. Here, the function is invoked using the form *module name.function name()* as shown below:

eg: import math

    print math.sin(math.pi/2)   # here math is the module name and sin() is the function name.

2. Here, another name is given to the module in the form import...........as ....

eg: import math as m

    print m.sin(m.pi/2)   #here another name is given to the math module as *m*

*3.H*ere, the module is imported as local,

eg: from math import sin   #here sin function is imported as local function

    print sin(pi/2)

4.Here, all functions are enabled from the module

eg: from math import*   # * indicates to enable all functions from the math module.i,e.,Here math module is imported as Global function.

    print sin(pi/2)

In the third and fourth cases, we need not type the module name every time.

But,there could be a trouble if two modules imported contains a function with same name. For eg. The sin() function from the numpy module is capable of handling a list argument. If we import math module in the same program, it will replace sin() from numpy. This causes an error because the sin() function can accept only a numeric type argument.

Eg:

```
from numpy import*
x=[0.10,0.2,0.3,0.4]
print sin(x)
from math import*
print sin(x)
```

verify the o/p

Packages:

Packages are used for organizing multiple modules. The module name A.B designates a submodule named B in a package named A. The concept is demonstrated in the following eg.

```
import numpy
print numpy.random.normal()   #here the form is A.B.C, C is the function name
import scipy.special
print scipy.special.j0(.1)
..................
................
```

in this example *random* is a module inside a package *Numpy.* Similarly *special i*s module inside a package *scipy.* We use both of them in the *package.module.function()* format

FILE INPUT AND OUT:

Files are used to store data and program for later use. The program creates a new file named 'myfile.txt'(any existing file with the same name will be deleted) and writes a *string* to it. The file is closed and then reopened for reading data. The relevant functions are open,write,read and close.

```
>>>>f=open('myfile.txt','w') #  (here , 'w' is the write mode.)
>>>f.write('hello world')
>>>f.close()
```

to read the file:

```
>>>f=open('myfile.txt','r')  #(here ,'r' is the read mode)
>>>print f.read()
>>>f.close()
```

check the output

PICKLE MODULE

Strings can easily be written to and read from a file. Numbers take a bit more effort, since the read() method only returns strings,which will have to be converted into a number explicitly. However, when you want to save and restore data types like lists, dictionaries,... , things get a lot more complicated. Rather than have the user constantly writing and debugging code to save complicated data type, Python provides a standard module called *pickle.*

Eg:

```
>>>import pickle
>>>a=1234
>>>b='physics'
```

```
>>>c=[5,2,3]
>>>f=open('myfile.pck',w')
>>>pickle.dump(a,f)
>>>pickle.dump(b,f)
>>>pickle.dump(c,f)
>>>>f.close()
```

To read the file:

```
>>>import pickle
>>>f=open('myfile.pck','r')
>>>a=pickle.load(f)
>>>b=pickle.load(f)
>>>>c=pickle.load(f)
>>>>print a*2,b,c
>>>f.close()
```