Syllabus

- Introduction to programming in python ➡ Introduction to programming, constants, variables and data types, dynamical typing, operators and expressions, modules, I/O statements, file handling, iterables, compound statements, indentation in python, the if-elif-else block, for and while loops, nested compound statements.
- Programs ➡ (a) Elementary calculations with different type of data e.g., area & volume of regular shapes using formula. Creation and handling one dimensional array. Sum and average of a list of numbers stored in array, finding the largest and lowest number from a list, swapping two data in a list, sorting of numbers in an array using bubble sort, insertion sort method. Calculation of term value in a series and finding the other terms with a seed (value of particular term) and calculation of different quantities with series. Convergence & accuracy of series. Introduction of three dimensional array. Simple calculations of matrices e.g., addition, subtraction, multiplication.

  (b) Curve fitting, Least square fit, Goodness of fit, standard deviation,

  i. Ohms law to calculate R,

  ii. Hooke's law to calculate spring constant

Introduction to Programming in Python

- Python is a scripting language, one of the most handy, easy-to-implement & open-source languages available. Unlike Fortran 77, 90/95 or C (high level), it directly interacts with the interpreter on every sentence executed on the command prompt (>>) [Python REPL-interpreter]. We'll however learn to write standard programs (py-scripts) in a file (suitably_choiced_name.py) & then execute it without compilation.

- Name REPL (Read-Evaluate-Print-Loops) is because it (a) reads what a user types, (b) evaluates what it reads, (c) prints out the return value after evaluation, and (d) loops back and does it all over again.

- Object oriented programming: deals with computable data as an object on which different methods act to produce a desired result. Its nature is defined as its properties. These properties can be probed by functions, which are called methods.

Introduction to Programming in Python

Field of Study	Python Module
Scientific Computation	numpy, scipy, sympy
Plotting/Visualization	matplotlib
Image Processing	scikit-image
Graphic User Interface (GUI)	pyQT
Statistics	pandas
Game Development	PyGame
Networking	networkx
Cryptography	pyOpenSSL
Database	SQLAlchemy
Language Processing	nltk
Testing	nose
HTML/XML parsing	BeautifulSoup
Machine Learning	scikit-learn, tensorflow

Introduction to Programming in Python

- Create a directory of your Group. This is where throughout the course your python scripts (codes) will be stored.
- Open **IDLE** editor & create a new file, say, hello.py.
  Within it, write: **print('Hello World')** Save & Run the code and welcome yourself !!
- Getting out of terminal : exit() or Ctrl + D.
- print ('Hello World')
  print 'I am a first year UG student'
  print "I'm new to Python Programming"
  print 'It is my first' + ' ' + 'Python code'
- ...code fragment... # Single line comments are commented out
  """
  Multiple line comment can be
  commented out like this.
  """

Introduction to Programming in Python

Syntax, Variables, Numbers, Operators:

Variables ➔ Name of a variable (or any identifier such as class, function, module or other object) can be anything combined with alphabets, letters & some symbols in the keyboard (except @, $, % etc.), beginning with a letter (the upper case or lower case letters from A to Z). But the names cannot be the words given in the following table. Those are Python keywords (in lower case letters) reserved for the use by system. Also, names are case sensitive. Suppose, we define a variable as 'ABC' and later we call as 'Abc', then it will not work.

Reserved words ➔ Using the module keyword, you can obtain the list of keywords.

import keyword; print ("Python keywords:", keyword.kwlist)

'Python keywords:', ['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

Introduction to Programming in Python

Syntax, Variables, Numbers, Operators

- Operators:

=	v = a + b	
+=	v = v + b	
-=	v = v – b	
/=	v = v / a	
//=	v = v // a	(roundoff)
*=	v = v * a	
**=	v = v ** a	(to the power)
%=	v = v % a	(remainder)

For **a=10, b=2**, Addition: **a+b = 12**, Substraction: **a–b = 8**, Multiplication: **a*b = 20**, Division: **a/b = 5**, Modulus [Remainder, when a is divided by b] **a%b = 0**, Power or exponent **a**b = 100**, Rounding off, e.g. if **a=100, b=3, 100/3= 33** also **100//3=33** but **100.0/3=33.3333**.....whereas **100.0//3 = 33.0**

I/O (Input/Output) Statements

```
a=input()  # Wait for the input value of 'a'.
a=input("Enter a: \n")
a, b, c = input()  # For many inputs.
```

- Multiple assignments within same statement: >> **a = b = c = 10**
- Order of usage: **PEMDAS (Parenthesis → Exponents → Multiplication → Division → Addition → Substracton.**

Introduction to Programming in Python

Comparison Operators:

Operator Symbol	Operator Meaning	Example
==	equal to	1==1 is True, 1==2 is False
!=	not equal to	1!=1 is False, 1!=2 is True
<>	not equal to	1<>1 is False, 1<>2 is True
<	less than	1<2 is True, 2<1 is False
>	greater than	1>2 is False, 2>1 is True
<=	less than equal to	1<=1 is True, 1<=2 is True
>=	greater than equal to	1>=1 is True, 1>=2 is False

```
>> not True
False
>> a = True; b = False; a and b
False
>> a or b : True
```

```
>> 1>=2 == 2>=1
False
>> 2>=1 == 1<=2
True
```

```
>> False > True
False
>> True > False
True
```

Introduction to Programming in Python

Array & List ➔ A list data type stores a sequence of values. All elements of list can be accessed by their index, but individual list elements can belong to any data type. Array on the other hand stores only numeric value & is not built in python interpreter ➔ need module "numpy" (Semester-3).

We can make a list of numbers or names or anything mixed, in the following way:

```
>>> X = [3, 2, 4, 1, 5, 0]                    # List of numbers
>>> Y = ["AKB", "MC", "SSB", "PD", "AD"]      # List of names
>>> Z = ["Good", 10, "Bad", 50]               # Mixed list
```

Display List ➔

```
>>> list(X)
[3, 2, 4, 1, 5, 0]

>>> list(Y)
"AKB", "MC", "SSB", "PD", "AD"
```

- To view, on the command prompt, **>>>X** Or **>>>print X** both works.
- Note, if we write **>>>list()**, we get back an empty list []. Also, while writing, the amount of gap between 'list' and the parenthesis () does not matter. You may write >>>list() & get back the same response.

Introduction to Programming in Python

>>> X+Y # sum of two lists (ans: **[2,3,4,1,5,0,'AKB','MC','SSB','PD','AD']**)

>>> X*3 # repeat 3 times (ans: **[2,3,4,1,5,0,2,3,4,1,5,0,2,3,4,1,5,0]**)

>>> len(X) # length of list (ans: **6**); **>>> sum(X)** # adding all element (ans: **15**)

>>> max(X) # maximum of X (ans: **5**), **>>> min(X)** # minimum of X (ans: **0**)

>>> X.index(max(X)) # Location of the maximum in list 'X' (ans: **4**)

>>> X.index(3) # Location of '3' in the list 'X' (ans: **1**)

- Positions are counted from left (like C programming): 0, 1, 2, 3,…

Numbers

int : Integers with +ive or -ive sign : Examples: 1245, -234 etc

long : Long Integers of unlimited size. Examples: 1245L

float : Floating point real numbers with a decimal point. Examples: 0.0, 2.24, -3.1e-10

complex : Complex numbers of the form a+bj, j = √-1, Examples: 1.2j, 1+2j, 1.2e-10j

Type conversion

>>> X = 15.56; int(X) # returns 15.

>>> X = 15; float(X) # returns 15.0

Introduction to Programming in Python

Playing with List

```
>>> X.append(10); list(X)          # Adding '10' at the right end of the list 'X' (ans: 2,3,4,1,5,0,10])
>>> Y.append('RM'); list(Y)        # Adding 'RM' at the right end of the list 'Y' (ans:
                                   ['AKB','MC','SSB','PD','AD','RM'])
>>> mean = sum(X)/len(X)           # Mean value from a list of numbers (ans: 15/6=2.5~2, as it
                                   is integer).
>>> mean=float(sum(X))/len(X)      # type-casting to float / make any entry in list float, say, 1.0.
>>> M = [3, ['a', -3.0, 5] ]       # (list within list) nested lists (for entering matrices)
>>> M[1][2]                        # returns 5
>>> range(5) or L = list(range(5))  # generate numbers from 0 to 4 [0,1,2,3,4]
>>> range(1,10)                    # returns [0,1,2,3,4,5,6,7,8,9]
>>> range(1, 10, 2)                # numbers from 1 to 9 with stride 2 [1, 3, 5, 7, 9].
>>> L=list(range(17,29,4))         # returns [17, 21, 25]
```

Introduction to Programming in Python

Slicing Slicing a list between i and j creates a new list containing the elements starting at index i and ending just before j. L[i : j] means create a list by taking all elements from L starting at L[i] until L[j-1] .

```
>>> a = list( [2, 3, 1, 4, 5, 6, 9, 10, 1] ); len(a)   # returns length of a is 9.
>>> a[:]     # returns entire list.
>>> a[1:]   # returns [3, 1, 4, 5, 6, 9, 10, 1]
>>> a[:1]   # returns 2
>>> a[3 : 6]  # returns [4, 5, 6]
```

Python allows the use of negative indexes for counting from the right. Element a[-1] is the last element in the list a.

- → a[i:] amounts to taking all elements except the i first ones,
- → a[:i] amounts to taking the first i elements,
- → a[-i:] amounts to taking the last i elements,
- → a[:-i] amounts to taking all elements except the i last ones.

Introduction to Programming in Python

$a[2:5]$	0	1	2	3	4	5	...	−1
$a[2:]$	0	1	2	3	...	−3	−2	−1
$a[:2]$	0	1	2	3	...	−3	−2	−1
$a[2:-1]$	0	1	2	3	...	−3	−2	−1
$a[-4:-1]$	0	1	...	−5	−4	−3	−2	−1
$a[:-2]$	0	1	2	3	...	−3		−1
$a[-2:]$	0	1	2	3	...	−3	−2	−1

Introduction to Programming in Python

Strides Length of the step from one index to the other (default stride is 1).

```
>>> L = list(range(100))   # generate 0 to 99
>>> L[:10:2]               # [0, 2, 4, 6, 8]
>>> L[::20]                # [0, 20, 40, 60, 80]
>>> L[10:20:3]             # [10, 13, 16, 19]
>>> L[20:10:-3]            # [20, 17, 14, 11] (negative stride).
```

- We can create a new list that is reversed using a negative stride →

```
>>> L = [1, 2, 3]
>>> R = L[::-1]            # R = [3, 2, 1]
```

Altering lists Insertion, deletion of elements & list concatenation. With slicing notation, list insertion & deletion is done, deletion is just replacing a part of a list by an empty list []

```
>>> L = ['a', 1, 2, 3, 4]
>>> L[2:3] = []            # ['a', 1, 3, 4]
>>> L[3:] = []             # ['a', 1, 3]
>>> L[1:1] = [1000, 2000]  # ['a', 1000, 2000, 1, 3]
```

Introduction to Programming in Python

- Two lists are concatenated by the plus operator + : >>> L = [1, -17];
  M = [-23.5, 18.3, 5.0]; L + M # returns [1, -17, -23.5, 18.3, 5.0]
- Concatenating a list n times with itself using multiplication operator * :
  >>> n = 3; n * [1.,17,3] # returns [1., 17, 3, 1., 17, 3, 1., 17, 3]
  >>> [0] * 5 # returns [0,0,0,0,0]

list.append(x) → Add x to end of the list.	L = [0, 1, 2, 3, 4] L.append(5) # [0, 1, 2, 3, 4, 5]
list.remove(x) → Remove first item from list with x.	L.remove(0) # [1, 2, 3, 4, 5]
list.insert(i,x) → Insert x at position i.	L.insert(0,0) # [0, 1, 2, 3, 4, 5]
list.count(x) → Number of times x appears in list.	L.count(2) # 1
list.sort() → Sort items of the list.	L.sort() # [0, 1, 2, 3, 4, 5]
list.reverse() → Reverse the elements of the list.	L.reverse() # [5, 4, 3, 2, 1, 0]
list.pop() → Remove last element of the list.	L.pop() # [0, 1, 2, 3, 4] L.pop() # [0, 1, 2, 3]

Introduction to Programming in Python

Tuples A tuple is an immutable list (cannot be modified). A tuple is just a comma-separated sequence of objects (a list without brackets or encloses a tuple in a pair of parentheses).

```
my_tuple = 1, 2, 3      # Our first tuple
my_tuple = (1, 2, 3)    # Same as previous
my_tuple = 1, 2, 3,     # Same as previous
len(my_tuple)           # 3, same as for lists
my_tuple[0] = 'a'       # error! Tuples are immutable
```

The comma indicates that the object is a tuple:

```
singleton = 1,          # Note the comma
len(singleton)          # 1
```

- Tuples are useful when a group of values goes together, e.g. they are used to return multiple values from functions. One may assign several variables at once by unpacking a list or tuple:

```
a, b = 0, 1     # a gets 0 and b gets 1
a, b = [0, 1]   # exactly the same effect
(a, b) = 0, 1   # same
[a,b] = [0,1]   # same thing
```

Introduction to Programming in Python

Array An array object strores a group of elements (values) of same datatype. Know more from ">>> help()" & then "help>>> array". To import array module, either, "import array" or "from array import *".

```
from array import *
a = array('i', [5, 6, -7, 8])        # i for signed integer, f for floating point (single dimension)
arr = array('d', [1.5, -2.2, 3.0, 5.75]  # d for double precision
m = array('d', [1.5, -2.2, 3.0, 5.75]   # d for double precision
```

Loops To execute a set of command repeatedly, in python "for", "while" loop is used.

For loop	While loop

sum=0
for x in range(begin, end, step):
  print "We are within a loop"
  sum += 1
print 'Sum of ', begin, ' to ', end, ' is = ', sum
	sum=0; n=5;
	while sum<n:
	  print sum
	  sum += 1
	print 'Summing 1 ', n-1, ' times = ', sum

Introduction to Programming in Python

Decision making (Logical statements) If, If … else, If … elif ...else, Nested if ...are used for logical decision making.

If: if x==10: print ‘x = ’, x

OR

```
if x==10:
    print “value of x is 10”
```

If-else: if x==10:

```
        print ‘x is 10’
    else:
        print ‘x is not 10’
```

If-elif-else:

```
a, b, c = input(‘Enter a,b,c: \n’)
x = b*b - 4*a*c
if x<0:
    print 'x is negative.'
elif x>0:
    print 'x is positive.'
else:
    print 'x is zero.'
```