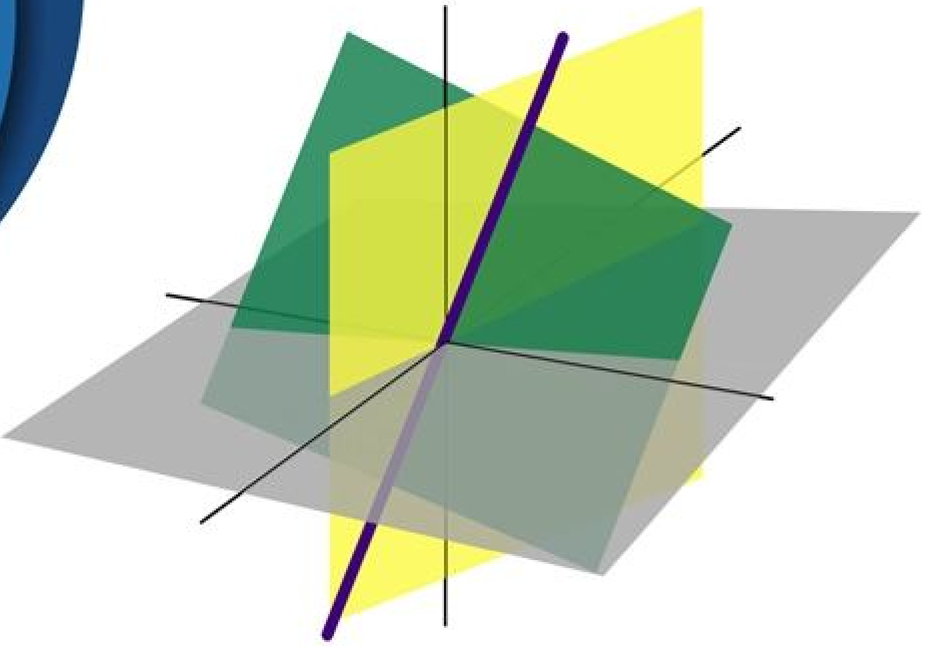
Linear Algebra Using Python

Archana Jadhav
Nandani Sakhare



Himalaya Publishing House
ISO 9001:2008 CERTIFIED

Linear Algebra Using Python

(As per the New Syllabus 2017-18 of Mumbai University for S.Y.B.Sc. (Computer Science), Semester – IV)

Archana Jadhav
MCS (Pune University)
Coordinator, Dept. of Computer Science,
N.G. Acharya & D.K. Marathe College,
Chembur, Mumbai.

Nandani Sakhare
M.Sc. (Mathematics), MBA (HR) and
B.Ed. (Science)
Assistant Professor, Dept. of Computer Science,
N.G. Acharya & D.K. Marathe College,
Chembur, Mumbai.

Himalaya Publishing House
ISO 9001:2008 CERTIFIED

First Edition : 2018

Published by	:	Mrs. Meena Pandey for **Himalaya Publishing House Pvt. Ltd.,** "Ramdoot", Dr. Bhalerao Marg, Girgaon, Mumbai - 400 004. **Phone:** 022-23860170, 23863863; **Fax:** 022-23877178 **E-mail:** himpub@vsnl.com; **Website:** www.himpub.com
Branch Offices	:	
New Delhi	:	"Pooja Apartments", 4-B, Murari Lal Street, Ansari Road, Darya Ganj, New Delhi - 110 002. Phone: 011-23270392, 23278631; Fax: 011-23256286
Nagpur	:	Kundanlal Chandak Industrial Estate, Ghat Road, Nagpur - 440 018. Phone: 0712-2738731, 3296733; Telefax: 0712-2721216
Bengaluru	:	Plot No. 91-33, 2nd Main Road, Seshadripuram, Behind Nataraja Theatre, Bengaluru - 560 020. Phone: 080-41138821; Mobile: 09379847017, 09379847005
Hyderabad	:	No. 3-4-184, Lingampally, Besides Raghavendra Swamy Matham, Kachiguda, Hyderabad - 500 027. Phone: 040-27560041, 27550139
Chennai	:	New No. 48/2, Old No. 28/2, Ground Floor, Sarangapani Street, T. Nagar, Chennai - 600 012. Mobile: 09380460419
Pune	:	"Laksha" Apartment, First Floor, No. 527, Mehunpura, Shaniwarpeth (Near Prabhat Theatre), Pune - 411 030. Phone: 020-24496323, 24496333; Mobile: 09370579333
Lucknow	:	House No. 731, Shekhupura Colony, Near B.D. Convent School, Aliganj, Lucknow - 226 022. Phone: 0522-4012353; Mobile: 09307501549
Ahmedabad	:	114, "SHAIL", 1st Floor, Opp. Madhu Sudan House, C.G. Road, Navrang Pura, Ahmedabad - 380 009. Phone: 079-26560126; Mobile: 09377088847
Ernakulam	:	39/176 (New No. 60/251) 1st Floor, Karikkamuri Road, Ernakulam, Kochi - 682 011. Phone: 0484-2378012, 2378016; Mobile: 09387122121
Bhubaneswar	:	Plot No. 214/1342, Budheswari Colony, Behind Durga Mandap, Bhubaneswar - 751 006. Phone: 0674-2575129; Mobile: 09338746007
Kolkata	:	108/4, Beliaghata Main Road, Near ID Hospital, Opp. SBI Bank, Kolkata - 700 010. Phone: 033-32449649; Mobile: 07439040301
DTP by	:	Rakhi
Printed at	:	M/s Sri Sai Art Printer, Hyderabad. On behalf of HPH.

Dedication

I would like to dedicate this book to my parents Late Mr. Vasant Deshmukh and Smt. Nirmala Deshmukh for their support in every step of my life and husband Mr. Sunil Jadhav for his encouragement and inspiration.

Archana Jadhav

I dedicate this book to my father Late Mr. Prabhakar Motghare, Mother Smt. Urmila Motghare and Husband Mr. Vikram Sakhare for their never-ending care and support.

Nandani Sakhare

We would like to express our sincere thanks to Mr. S.K. Srivastava of Himalaya Publishing House for giving us the opportunity to write this book. We express our gratitude towards Mrs. Kiran Gurbani for her support and guidance.

Preface

It gives us great pleasure to present a book ***"Linear Algebra using Python"*** for S.Y.B.Sc. (Computer Science), Semester IV. This book is based on new syllabus prescribed by Mumbai University from 2017 – 18. The main topics have been divided into subtopics for easy understanding. A number of solved examples have been included in the book with their implementation in Python.

We hope this book will fulfil all your needs for the subject. As a step towards betterment and improvement of the book, suggestions from teachers and students are welcome.

Finally, we would like to acknowledge our sincere respect and gratitude to Kiran Gurbani (Head of Computer Science and IT Department, R.K. Talreja College, Ulhasnagar) for reviewing this book thoroughly and providing an environment which stimulates new thinking and innovations and her support which helped us for bringing out this book in time.

Please send correspondence to archana.sjd@gmail.com, nandani1002@gmail.com.

We are thankful to Shri S.K. Srivastava and staff members of Himalaya Publishing House for their cooperation.

Authors

Syllabus

Linear Algebra using Python

Course Code: USCS405

Objectives:

To offer the learner the relevant linear algebra concepts through computer science applications.

Expected Learning Outcomes:

1. Appreciate the relevance of linear algebra in the field of computer science.
2. Understand the concepts through program implementation.
3. Instill a computational thinking while learning linear algebra.

Unit No.	Modules/Units
Unit 1	**Field:** Introduction to complex numbers, numbers in Python, Abstracting over fields, Playing with GF(2), Vector Space: Vectors are functions, Vector addition, Scalar-vector multiplication, Combining vector addition and scalar multiplication, Dictionary-based representations of vectors, Dot-product, Solving a triangular system of linear equations. Linear combination, Span, The geometry of sets of vectors, Vector spaces, Linear systems, homogeneous and otherwise
Unit 2	**Matrix:** Matrices as vectors, Transpose, Matrix-vector and vector-matrix multiplication in terms of linear combinations, Matrix-vector multiplication in terms of dot-products, Null space, Computing sparse matrix-vector product, Linear functions, Matrix-matrix multiplication, Inner product and outer product, From function inverse to matrix inverse **Basis:** Coordinate systems, Two greedy algorithms for finding a set of generators, Minimum Spanning Forest and GF(2), Linear dependence, Basis, Unique representation, Change of basis, first look, Computational problems involving finding a basis **Dimension:** Dimension and rank, Direct sum, Dimension and linear functions, The annihilator
Unit 3	**Gaussian elimination:** Echelon form, Gaussian elimination over GF(2), Solving a matrix-vector equation using Gaussian elimination, Finding a basis for the null space, Factoring integers **Inner Product:** The inner product for vectors over the reals, Orthogonality **Orthogonalization:** Projection orthogonal to multiple vectors, Projecting orthogonal to mutually orthogonal vectors, Building an orthogonal set of generators, Orthogonal complement **Eigenvector:** Modeling discrete dynamic processes, Diagonalization of the Fibonacci matrix, Eigenvalues and eigenvectors, Coordinate representation in terms of eigenvectors, The Internet worm, Existence of eigenvalues, Markov chains, Modeling a web surfer: PageRank

Question Paper Pattern

(2½ Hours) **[Total Marks: 75]**

N.B. 1. All questions are compulsory.

2. Figures to the right indicate marks.

Q.1	**Attempt All (Each of 5Marks)**	**(15M)**
(a)	Select correct answer from the following:	(5)
(b)	Fill in the blanks:	(5)
(c)	Define the following:	(5)
Q.2 (a) (b) (c) (d) (e) (f)	**Attempt the following (Any THREE) from Unit I**	**(15M)**
Q.3 (a) (b) (c) (d) (e) (f)	**Attempt the following (Any THREE) from Unit II**	**(15M)**
Q.4 (a) (b) (c) (d) (e) (f)	**Attempt the following (Any THREE) from Unit III**	**(15M)**
Q.5 (a) (b) (c) (d) (e)	**Attempt the following (Any THREE) from Unit I, II & III**	**(15M)**

Contents

Unit I

Chapter 1 Field

Structure:

1.1 LINEAR ALGEBRA IMPLEMENTATION USING PYTHON

Python is a great general-purpose programming language on its own. But the language has no built-in support for linear algebra.

With the help of few popular libraries like Numpy, scipy and matplotlib it provides a powerful environment for scientific computing. Num Py is the best option for complex numerical linear algebra implementation. Matplotlib is a Python 2D plotting library which produces publication quality figures and graphs in a variety of formats

Even without the help of these packages we can implement Linear Algebra concepts by using collection data types of python. Like vectors can be implemented by lists and matrices can be implemented with the help of nested lists.

In this book the implementation is shown by both the ways. Let us have a brief overview of Python collection data type.

1.1.1 Collection Data Types in Python

Python provides data type called as collections for grouping together multiple values.

Sets, Lists, Tuples and Dictionary are some of the collections data types.

1. Sets.

A set is an unordered collection which cannot have duplicate value, It is usually used to build sequence of unique items. It is represented by curly braces{}.

```
>>> {1+1,2, "a"}
{'a', 2}
>>> {2, 0, 3}
{0, 2, 3}
```

Note that duplicates are eliminated from the output and the order in which elements are printed does not necessarily same as the order of the input elements.

The empty set is represented by set().

Operations on Set

- **The cardinality of a set S-**

  In maths we write |S| is obtained using the procedure **len()**.

  ```
  >>> len({'a', 'b', 'c', 'd', 'a'})
  4
  ```

- **Summing**

  The sum of elements of set is obtained using the procedure **sum().**

  ```
  >>> sum({1,3,3})
  4
  >>> sum({1,3,3}, 10)
  14
  ```

  Here the second argument is the value to start with for summation.

- **Set membership test**

  To test the element is a member of set or not we can use **in** operator and the **not in** operator. If S is a set, x in S is a Boolean expression which evaluates to True if the value of x is a member of the set S, and False otherwise**.**

  Not in expression is just the opposite of **in**

  ```
  >>> S={1,2,3,5}
  >>> 2 in S
  True
  >>> 2 not in S
  True
  ```

- **Set union and intersection**

  The union of two sets S1 and S2 is a new set that contains every member of s1 or member of s2 or member of both the sets. Vertical bar '|' is used as the union operator:

  ```
  >>> {1, 2} | {2, 3, 4}
  {1, 2, 3, 4}
  ```

  The intersection of S1 and S2 is a new set that contains only those members which are present in both the sets. Ampersand '&' is used as the intersection operator:

  ```
  >>> {1, 2, 3} & {2, 3, 4}
  {2, 3}
  ```

- **Mutating a set**

  Sets are mutable data type. The elements can be added and removed using add and remove methods:

  ```
  >> S={2, 3, 5}
  >>> S.add(4)
  >>> S.remove(2)
  >>> S
  {3, 4, 5}
  ```

  You can add all the elements of another collection to a set by using update () method

  ```
  >>> S.update({4, 5, 6})
  >>> S
  { 3, 4, 5, 6}
  ```

- **Set comprehensions**

  Comprehensions are a special notation for building up collections from other collections, modifying and filtering them in the process. It can be used to construct collections in a very natural, easy way, like a mathematician is used to do. It express complicated looping logic in a simple format. You can use union and intersect operators in comprehension

  For example:

  ```
  >>> {2*x for x in {1, 2, 3, 4} }
  {2, 4, 6, 8}
  ```

  At the time of execution Python iterates over the elements of set {1, 2, 3, 4}, binds the variable x to each element of set. Evaluate the expression 2*x in each iteration and construct the final set.

  Examples on comprehension

  List the square of all the members of both the sets

  ```
  >>>S={1,3,4}
  >>>S1={1,5}
  >>>{x*x for x in S | S1}
  {1, 9, 16, 25}
  ```

You can skip some of the values of set while iterating

List the square of only even members of set

```
>>>{x*x for x in S if x%2 == 0}
{16}
```

Double comprehension

You can write a comprehension that iterates over the Cartesian product of two sets:

```
>>>{x*y for x in {1,2,3} for y in {1,3,4}}
{1, 2, 3, 4, 6, 8, 9, 12}
```

This comprehension constructs the set of the products of every combination of x and y.

2. Lists

List is a mutable collection data type used to represents sequences of values. In a list, order is significant and repeated elements are allowed. List is represented by square bracket [].An empty list is represented by [].

```
>>>L=[]        # It will create empty list L.
>>> [1,1+1,3,2,3]
 [1, 2, 3, 2, 3]
```

A list can hold any type of values. It may contain simple values or collections like a

set or another list. However, a set cannot contain a list since lists are mutable.

```
>>> [[1,2,3],{2*2,5,6}, "abc",7]
[[1, 2, 3], {4, 5, 6}, 'abc', 7]
```

Operations on List

- **Compute Length**: Like Set the procedure len () is used to obtain the length of list,

  ```
  >>> len[[1,2,3],{2*2,5,6}, "abc",7])
  4
  ```

- **Summing** : Sum of elements of a list is computed using procedure sum()

  ```
  >>> sum([1,2,0,3,1,2,1])
  10
  >>> sum([1,2,0,3,1,2,1], -9)
  -1
  ```

  Here the second argument is the value to start with for summation.

- **List concatenation :** You can combine the elements of two or more lists using the + operator.

  ```
  >>> [1,2,3]+["X”,”Y”, "Z"]
  [1, 2, 3, ‘X’,’Y’,’Z’]
  ```

  You can even use procedure sum to concatenate collection of lists.

  ```
  >>> sum([ [1,2,3], [4,5,6], [7,8,9] ], [])
  [1, 2, 3, 4, 5, 6, 7, 8, 9]
  ```

use [] (empty list) as the second argument in procedure sum.

- **Membership Test** : Like set List also support in operator to test membership.

```
>>> 1 in [1, 2, 3]
True
```

- **Mutating list :** List data type supports Insert & append method for list mutation

```
>>> L=[1, 2, 3]
>>> L.append(4)
>>> L.insert(1,5)
>>> L
[1, 5, 2, 3, 4]
```

To remove element from list

```
>>> del(L[1])
>>> L
[1, 2, 3, 4]
```

- **List Retrival:** There are two ways to retrieve elements of list by indexing and by unpacking.

By Indexing

```
>>>mylist=[4, 5, 6, 7, 8]
```

The index value of the first element of the list is 0.

```
>>> mylist[1]
5
```

By Slices You can retrieve consecutive subsequence of elements from the list by using slice operator [i:j] this I and specifies the range of index value.

```
>>>mylist[1:3]
[5,6]
>>>mylist[:2]
[4, 5]
>>>mylist[3:]
[7, 8]
```

In slicing to skip the elements you can use a colon-separated triple a:b:c.The slice will include every cth element

```
>>>mylistL[::2]
[4, 6, 8]
```

By unpacking

Instead of assigning a list to a single variable as in mylist =[4,5,6], one can assign to a list of variables:

```
>>> [x,y,z] = [4, 5, 6]
>>> x
4
```

```
>>> y
5
```

Unpacking is used in list comprehensions:

```
>>> L1 = [[1, 1],[2, 4],[3, 9]]
>>> [y for [x,y] in L1]
[1, 4, 9]
```

- **List comprehensions**

  Like sets You can perform comprehension on lists

  ```
  >>> [ 2*x for x in [2, 1, 3, 4, 5] ]
  [4, 2, 6, 8, 10]
  ```

  Here is an example of a list comprehension over two lists.

  ```
  >>> [ x*y for x in [1, 2, 3] for y in [10, 20, 30] ]
  [10, 20, 30, 20, 40, 60, 30, 60, 90]
  ```

  The resulting list is a Cartesian product of both the lists

  Use reversed keyword to iterate in reverse order.

  ```
  >>> [x*x for x in reversed([1, 2, 3])]
  [9, 4, 1]
  ```

3. Tuples

Tuple is an ordered sequence of elements like list but it is immutable. The tuples are represented by parenthesis ().

```
>>> (1, 1+1, 3)
(1, 2, 3)
>>> {0, (1, 2)} | {(3, 4, 5)}
{(1, 2), 0, (3, 4, 5)}
```

Operations on Tuple

- You can obtaining elements of a tuple by indexing and unpacking

  Indexing

  ```
  >>> mytuple = ("cs", "it", "BMM")
  >>> mytuple[1]
  'it'
  ```

  Unpacking

  Here is an example of top-level variable assignment:

  ```
  >>> (a,b) = (1,5)
  >>> a
  1
  ```

- You can do comprehension on tuples like Sets and Lists

```
>>> [2*x for x in (1, 2, 3)]
[2, 4, 6]
```

4. Zip

A zip is collections in python which is constructed from other collections all of the same length. Each element of the zip is a tuple consisting of one element from each of the input collections.

```
>>> list(zip([2, 4, 6],[1, 3, 5]))
[(2, 1), (4, 3), (6, 5)]
```

You can construct zip by procedure set

```
>>> characters = ['Neo', 'Morpheus', 'Trinity']
>>> actors = ['Keanu', 'Laurence', 'Carrie-Anne']
>>> set(zip(characters, actors))
{('Morpheus', 'Laurence'), ('Neo', 'Keanu'), ('Trinity', 'Carrie-Anne')}
```

5. Dictionaries

A dictionary is a set of key-value pairs. We will often have occasion to use functions with finite domains. Dictionaries are suitable for representing such functions.

The syntax for creating dictionary is {key: value}

For example the function f that maps each letter in the alphabet to its rank in the alphabet could be written as:

```
f={'A':0, 'B':1, 'C':2, 'D':3, 'E':4, 'F':5, 'G':6, 'H':7, 'I':8,'J':9, 'K':10, 'L':11, 'M':12, 'N':13, 'O':14, 'P':15, 'Q':16,'R':17, 'S':18, 'T':19, 'U':20, 'V':21, 'W':22, 'X':23, 'Y':24,'Z':25}
```

Points to Remember:-

- The order of the key-value pairs is irrelevant.
- The key in a dictionary corresponds to only one value. If multiple values are assigned for the same key, only one value will be associated with that key.
- The keys will mostly be integers, strings, or tuples of integers and strings.
- The keys and values can be specified with expressions.

```
>>> {2+1:'thr'+'ee', 2*2:'fo'+'ur'}
{3: 'three', 4: 'four'}
```

Operations on Dictionary

- You can obtain elements of a dictionary by indexing on key

```
>>> {4:"four", 3:'three'}[4]
'four'
>>> mydict = {'dbms':'Oracle', 'webapp':'PHP','OOPs':'Java'}
>>> mydict['OOPS']
'Java'
```

- **Membership Testing**

  You can check whether a key is present in a dictionary using the in operator

  ```
  >>> 'dbms' in mydict
  true
  >>> mydict['dbms'] if 'PRESENT' in mydict else 'NOT PRESENT'
  'PRESENT'
  ```

- **Mutating a Dictionary:**

  To update the content of dictionary access the dictionary with key and assign new value to dictionary.

  ```
  >>> mydict['dbms'] = 'MYSQL'
  >>> mydict
  {'dbms':'MYSQL', 'webapp':'PHP','OOPs':'Java'}
  ```

  To add new element in dictionary

  ```
  >>>mydict['stats']= 'RTOOL'
  >>> mydict
  {'dbms': 'MYSQL', 'webapp': 'PHP', 'OOPs': 'Java', 'stats': 'RTOOL'}
  ```

  To remove element from dictionary

  ```
  >>>del(mydict[OOPS])
  >>> mydict
  {'dbms': 'MYSQL', 'webapp': 'PHP', 'stats': 'RTOOL'}
  ```

- **Dictionary Comprehensions**

  Like set, lists and tuple you can use comprehension to construct dictionary

  ```
  >>> { key:value for (key,value) in [(1,1),(2,4),(3,9)] }
  {1: 1, 2: 4, 3: 9}
  >>> { x:x*x for x in [1,2,3]}
  {1: 1, 2: 4, 3: 9}
  ```

  You can retrieve keys and values of dictionary by using procedure keys() or values():

  ```
  >>> mydict.keys()
  dict_keys(['dbms', 'webapp', 'OOPs'])
  >>> mydict.values()
  dict_values(['Oracle', 'PHP', 'Java'])
  >>>
  ```

  These procedures can be used in comprehension also.

  ```
  >>> [2*e for e in {4:'x',3:'y'}.keys() ]
  [6, 8]
  >>> [e for e in {4:'x', 3:'y'}.values()]
  ['y', 'x']
  ```

1.1.2 Popular Numeric and Scientific Python Packages

- **numpy -** Numerical Python adds a fast, compact, multidimensional array facility to Python.
- **scipy -** is an open source library of scientific tools for Python. It supplements the popular NumPy module, gathering a variety of high level science and engineering modules together as a single package.
- **matplotlib-** is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- **Python Data Analysis Library (PANDAS) -** pandas is a library providing high-performance, easy-to-use data structures and data analysis tools for the Python.
- **FuncDesigner -** is Python module to rapidly build functions and get their derivatives via automatic differentiation.
- **OpenOpt -** a framework for numerical optimization and systems of linear/non-linear equations.
- **SpaceFuncs -** a tool for 2D, 3D, N-dimensional geometric modeling
- **Scientific Python -**Scientific Python is a collection of Python modules that are useful for scientific computing. In this collection you will find modules that cover basic geometry (vectors, tensors, transformations, vector and tensor fields), quaternions, automatic derivatives, (linear) interpolation, polynomials, elementary statistics, nonlinear least-squares fits, unit calculations.

Exercise 1.1

Test Your Collection Comprehension Skills.

1. Write a comprehension to construct a new set with odd values of set S={1, 2, 3, 4, 5}.
2. Write a comprehension to construct a new set S1 which members are the powers of 2 of member of set S={0, 1, 2, 3, 4} i.e S1={ 2^0, 2^1, 2^2, 2^3, 2^4}
3. Assume that S1 and S2 are the sets with the values S1 = {1, 2, 3, 4} and S2 = {3, 4, 5, 6}. Write a comprehension to construct a new set which have the unique members of both the sets.
4. Write a comprehension to retrieve positive numbers from list.
5. Write a comprehension to construct a list of odd numbers from 1 to 100.
6. From the lists L1= [10, 25, 40] and L2 = [1, 15, 20], write a comprehension to construct a list which elements are addition of elements of L1 and L2.
7. Without using comprehension construct a list

   New=[(0, 'A'), (1, 'B'), (2, 'C'), (3, 'D'), (4, 'E')] by using list L = ['A','B','C','D','E'].
8. Write a comprehension whose value is a dictionary. The keys should be the integers from 0 to 99 and the value corresponding to a key should be the square of the key.
9. Using the variables base=2 and digits=set(range(base)), write a dictionary comprehension that maps each integer between zero and seven to the list of three digits that represents that integer in base 2. That is, the value should be

   {0: [0, 0, 0], 1: [0, 0, 1], 2: [0, 1, 0], 3: [0, 1, 1],..., 7: [1, 1, 1]}

10. Suppose d is a dictionary that maps employee IDs to salaries. Suppose L is an n-element list whose ith element is the name of employee number i. Write a comprehension which maps employee name to salary. Test your comprehension with the following data:

    salary = {0:1000.0, 3:990, 1:1200.50}

    emp = ['Larry', 'Curly', '', 'Moe']

1.2 COMPLEX NUMBERS

This section deals with the fundamental definitions and operations involving complex numbers. Since the graphical operations of addition and subtractions of complex numbers are analogous to corresponding operations of vectors. Complex numbers are frequently applied to vector problems in Mathematics.

1.2.1 Introduction

In solving quadratic equations, we often get a square root of negative number as the part of the values of the roots for example if $x^2 - 2x + 2 = 0$. Then the roots are:

$$x = \frac{2\pm\sqrt{-4}}{2} = 1 \pm \frac{1}{2}\sqrt{-4}$$

The square roots of negative numbers are not possible. Hence a new type of number known as imaginary number is introduced to extract the square root of any negative number which is expressed in terms of $\sqrt{-1}$, known as imaginary unit and usually denoted by i.

The fundamental property of the symbol is $i^2 = -1$.To make the real and imaginary numbers part of one system, a new number is introduced called complex number represented by x + iy.

Definition: An expression of the form x + iy, where x and y are real numbers and i $=\sqrt{-1}$, is called a complex number which is generally denoted by z

i.e. z = x + iy,

where x and y are respectively called real part and imaginary part of z.

i.e. Re (z) = x and Im (z) = y

If x=0 the number is called purely imaginary and if y=0 then the number is called purely real.

1.2.2 Equality of Complex Numbers

Theorem 1: Two complex numbers are equal if their real and imaginary parts are equal.

Proof: Let $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$ are equal.

$x_1 + iy_1 = x_2 + iy_2$

$\therefore (x_1 - x_2) = i(y_1 - y_2)$

Squaring both the sides, we get

$(x_1 - x_2)^2 = -(y_1 - y_2)^2$

as $i^2 = -1$

$\therefore (x_1 - x_2)^2 + (y_1 - y_2)^2 = 0$

$\therefore x_1 = x_2$ and $y_1 = y_2$

e.g. if x + iy = 3 + 2i then x = 3 and y = 2

if x + iy = 2i then x = 0 and y = 2

Example1: If 2a + i (3b) = 6 + 3i, find a, b

By data 2a + i (3b) = 6+ 3i

$\therefore 2a = 6$ and 3b = 3

$\therefore$ a = 3 and b = 1

1.2.3 Operations on Complex Numbers

Arithmatic Operations

The sum, difference, product or quotient of any two complex numbers is a complex number.

1.	Addition	:	(a + ib) + (c + id) = (a + c) + i (b + d)
2.	Subtraction	:	(a + ib) - (c + id) = (a - c) + i(b - d)
3.	Multiplication	:	(a + ib) x (c + id) = ac – bd + i (ad + bc)
4.	Division	:	$\frac{(a+ib)}{(c+id)} = \frac{(a+ib)}{(c+id)} \cdot \frac{(c-id)}{(c-id)} = \frac{ac+bd}{c^2+d^2} + i\frac{bc-ad}{c^2+d^2}$

Example 1: Express $\frac{(1+i)(2+i)}{(3+i)}$ in the form a+ib

Solution: $\frac{(1+i)(2+i)}{(3+i)} = \frac{2+i+2i-1}{3+i} = \frac{1+3i}{3+i} = \frac{(1+3i)(3-i)}{(3+i)(3-i)}$

$= \frac{3-i+9i+3}{9+1} = \frac{6+8i}{10} = \frac{3}{5} + \frac{4}{5}i$

Example 2: Express $\frac{(6+i)(2-i)}{(4+3i)(1-2i)}$ in the form a + ib

Solution: $\frac{(6+i)(2-i)}{(4+3i)(1-2i)} = \frac{12+1+i(2-6)}{4+6+i(3-8)}$

$= \frac{13-4i}{10-5i} = \frac{(13-4i)(10+5i)}{(10-5i)(10+5i)}$

$= \frac{150+25i}{100+25} = \frac{6+i}{5} = \frac{6}{5} + \frac{1}{5}$ i

Square Root of a Complex Number

If z = a + ib then the $\sqrt{z}$ can be obtained by equating $\sqrt{a+ib}$ with a complex number x + iy i.e

$\sqrt{z}$ = x + iy

$\therefore \sqrt{a+ib} = x + iy$………..eq.1

Taking square both the sides

$\therefore (a+ib)$= $(x+iy)^2$

= x^2-y^2+2xy i

Equating real and imaginary part

a =x^2-y^2 and b = 2xy

By solving the above two equations, get the value of x and y and hence the result by substituting the values in equation 1

Example 1: Find the square root of 21-20 i

Solution: Suppose $\sqrt{21-20i}$ = x + iy

$\therefore 21-20\,i=(x+iy)^2$,

= x^2-y^2+ 2xy i

$\therefore\ x^2$-y^2 = 21and -20=2xy

x=$\frac{-10}{y}$

$\Rightarrow\left(\frac{-10}{y}\right)^2-y^2$=21

$\Rightarrow\frac{100}{y^2}$ -y^2=21

$\Rightarrow$100-y^4=21y^2

$\therefore y^4$+21y^2-100=0

$\Rightarrow y^4$+25y^2-4y^2-100=0

$y^2=25\ or y^2=-4$

$y=\mp$ 2i or y = $\pm$ 5

Y cannot be a real number hence $y=\mp$ 2i

And x = $\pm$ 5

$\therefore x+iy=\pm(5-2i)$.

Conjugate of Complex Number

If z = a + ib is a complex number then the number a-ib is called its conjugate and it is denoted by $\bar{z}$

$\therefore\ \bar{z}$ = a – ib

Example 1: Find the conjugate of z = 3 + 2i .

Solution: z = 3 + 2i

$\therefore\ \bar{z}=3-2i$

Example 2: Find the conjugate of z = -9- 2i .

Solution: z = -9 - 2i

$\therefore\ \bar{z}=-9+2i$

Exercise 1.2

1. Express the following in the form a + ib, where a and b are real numbers.

   (i) $\frac{2-3i}{4-i}$ (ii) $\frac{(3+4i)(2+i)}{(1+i)}$ (iii) $\frac{(3+2i)}{(4-5i)(2+i)}$ (iv) $\frac{(1+2i)^3}{(1+i)(2-i)}$ (v) $\frac{(2+3i)^2}{1+i}$

2. Find the square root of z=-5 +2i

3. Find modulus Principal argument of

   (i) -$\sqrt{3}-i$ (ii) $\frac{1+2i}{1-3i}$ (iii) $\frac{(1+i)^2}{1-i}$ (iv) 1- cos$\alpha+isin\alpha$ (v) 1+ i$\sqrt{3}$

Answers:

Q. 1: (i) $\frac{11}{17}+\frac{10}{17}$i (ii) $\frac{13}{2}+\frac{9}{2}$i (iii) $\frac{27}{205}+\frac{44}{205}i$ (iv) $-\frac{-7}{2}+\frac{i}{2}$ (v) $\frac{7}{2}+\frac{17}{2}$i

Q. 2: (i) $\pm(2+3i)$

Q. 3: (i) $2,\frac{-5\pi}{6}$ (ii) $\frac{1}{\sqrt{2}},\frac{3\pi}{4}$ (iii) $\sqrt{2},\frac{3\pi}{4}$ (iv) $2\sin\frac{\alpha}{2},\frac{\pi-\alpha}{2}$ (v) $2,\frac{\pi}{3}$

1.3 GEOMETRICAL REPRESENTATION OF COMPLEX NUMBER (ARGAND DIAGRAM)

Mathematician Argand represented a complex number in diagram known as Argand diagram. A complex number x+iy can be represented by a point P whose co-ordinates are (x, y).The axis of x is called the real axis and the axis of y is called the imaginary axis . The distance OP is the modulus and the angle(θ), OP makes with x-axis, is the argument of x + iy.

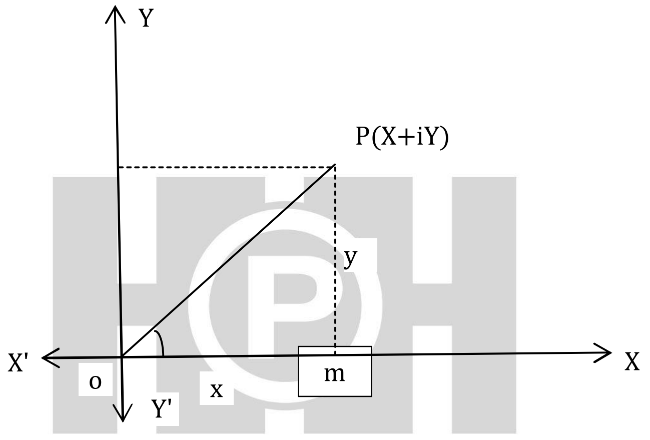


Fig. 1.1

If z = x + iy be a complex number and p(x,y) be a point in xy plane the PM=y and OM=x

Let OP be r then

$OP \ = \sqrt{OM^2+PM^2}$

r $=\sqrt{x^2+y^2}$

r is called modulus of complex number and denoted by $|z|$

$\therefore \ |z|=\sqrt{x^2+y^2}$

If OP makes an angle $\theta\ with$ positive x axis then tan$\theta=\frac{PM}{OM}=\frac{Y}{X}$

$\therefore \theta=\tan^{-1}\frac{y}{x}$

θ is called amplitude or argument of z and denoted by amp(z)

Let z= x+iy be a complex number, Putting x=rcos θ and y=r sin θ so that r $=\sqrt{x^2+y^2}$

$\therefore \cos\theta=\frac{x}{\sqrt{x^2+y^2}}and\sin\theta=\frac{y}{\sqrt{x^2+y^2}}$

$\therefore$ The complex number in polar form is r($\cos\theta+i\sin\theta$).

Example 1: Find the modulus and principal argument of the complex number $\frac{1+2i}{1-(1-i)^2}$.

Solution: Z $=\frac{1+2i}{1-(1-i)^2} = \frac{1+2i}{1-(1-1-2i)} = \frac{1+2i}{(1+2i)} = 1$

$\therefore$ z=1 + 0i

Since $Modulus of z is |z|=\sqrt{x^2+y^2}$

$\therefore\ |z| = \sqrt{1^2+0^2}$ =1 Ans

Principal Argument of z= $\frac{1+2i}{1-(1-i)^2} = 1 + 0\ i$

$\text{Amp(z)} = \theta = \tan^{-1}\frac{y}{x}$

$\therefore \theta = \tan^{-1}\frac{0}{1} = \theta = \tan^{-1} 0$=0° Ans

Example 2: Express $\frac{1+2i}{1-3i}$ in the Polar form .

Solution: $\frac{1+2i}{1-3i} = \frac{(1+2i)(1+3i)}{(1-3i)(1+3i)} = \frac{1-6+5i}{1+9} = \frac{-5+5i}{10} = -\frac{1}{2} + \frac{i}{2}$

$\therefore$ r $=\sqrt{\frac{-1}{2^2}^2 + \frac{1}{2^2}^2}$

$= \sqrt{\frac{1}{4}+\frac{1}{4}} = \sqrt{\frac{1}{2}} = \frac{1}{\sqrt{2}}$

Amp (z) $=\theta = \tan^{-1}(-1) = \frac{3\pi}{4}$

$\therefore$ z$=\frac{1}{\sqrt{2}}\left(\cos\frac{3\pi}{4} + i\sin\frac{3\pi}{4}\right)$

Ans

1.4 COMPLEX NUMBERS IN PYTHON

A complex number consists of an ordered pair of real floating-point numbers denoted by x + yj, where x is the real part and b is the imaginary part of the complex number. In Python complex number is one of the four numerical data type, which is represented in a format x+yJ or x+yj.

```
>>>type (1+2j)
<class 'complex'>
```

The type of a complex number is complex. The class complex is defined in Python to support Complex number data type. This class defines the procedures used in arithmetic operations on complex numbers. You can use the type as a constructor if you prefer:

```
>>> complex(2,3)
(2+3j)
```

The square root of -9, the imaginary number 3i, is written as 3j. Thus j plays the role of i.

```
>>>3j
3j
```

The square root of -1, the imaginary number i, is written 1j so as to avoid confusion with the variable j.

```
>>>1j
1j
```

Arithmetic Operations on Complex Numbers

The arithmetic operators +, -, *, /, and ** all work with complex numbers in Python.

- When you add two complex numbers, the real parts are added and the imaginary parts are added.

```
>>> (1+3j) + (10+20j)
(11+23j)
```

```
>>> x=1+3j
>>> (x-1)**2
(-9+0j)
```

Python considers the value (-9+0j) to be a complex number even though its imaginary part is zero.

- As in ordinary arithmetic, multiplication has precedence over addition; exponentiation has precedence over multiplication. So, by these precedence rules following expressions are evaluated as follows:

```
>>> 1+2j*3
(1+6j)
>>> 4*3j**2
(-36+0j)
```

- You can obtain the real and imaginary parts of a complex number using a dot notation.

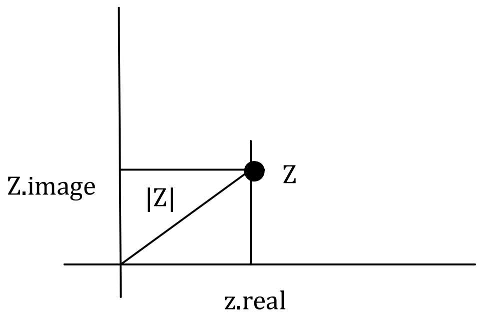
```
>>>x.real
1.0
>>>x.imag
3.0
```

- You can obtain conjugate of complex number by

```
>>>x.conjugate()
(1-3j)
```

1.5 PLAYING WITH C

As a complex number z consists of two ordinary numbers, z.real and z.imag, You can consider z as specifying a point, a location, in the plane (the complex plane).



|Z| The absolute value of a complex number

The absolute value of a complex number z i.e |z| is the distance from the origin to the corresponding point in the complex plane

By the Pythagorean Theorem, $|z|^2 = (z.real)^2 + (z.imag)^2$.

```
>>> abs(3+2j)
3.6055512754639896
>>>abs(1)
1.0
```

Let us plot the set of complex numbers S, for each location in the complex plane where we want a dot, we include the corresponding complex number in S.

Consider S = {3 + 3i, 4 + 3i, 2 + i, 2.5 + i, 3+ i, 3.25 + i}

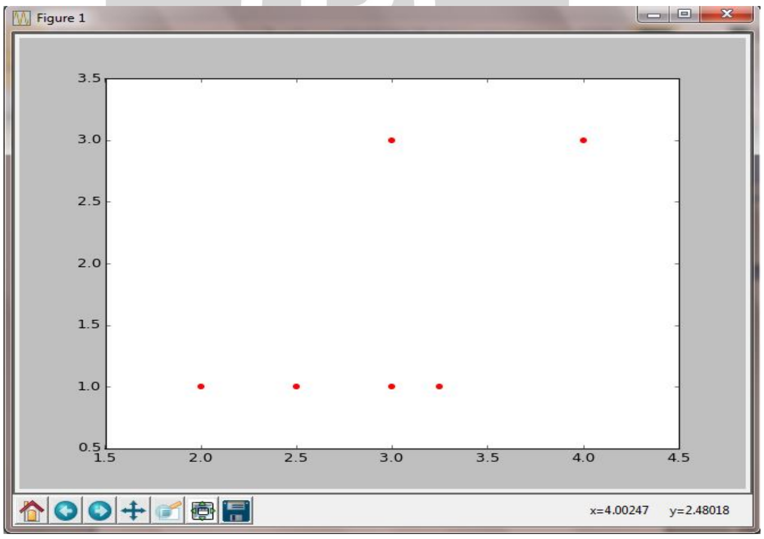


Fig. 1.1

Translation

Let us add a complex number 1 + 2i to each complex number Z in set S. We get a new set of complex numbers S1.Thus we obtained a new function ‘f ’ for this application.

i.e f (z) = 1 + 2i + z

This function increases each element z of S by 1 unit on real axis and by 2 units on imaginary axis. This results into the shift of graph by one unit to the right and 2 unit to the upward direction. Such that S1= {4 + 6i, 5 + 5i, 3 +3i, 3.5 +3i, 4+3i, 4.25 +3i} Refer Fig 1.3 for the shifted picture.

Therefore we generalize the formula for the transformation of a set of complex numbers as

f (z) = z_0 + z , z_0 is a complex number.

This transformation of numbers in S is called translation. It helps us to transform the image, anywhere in the complex plane.

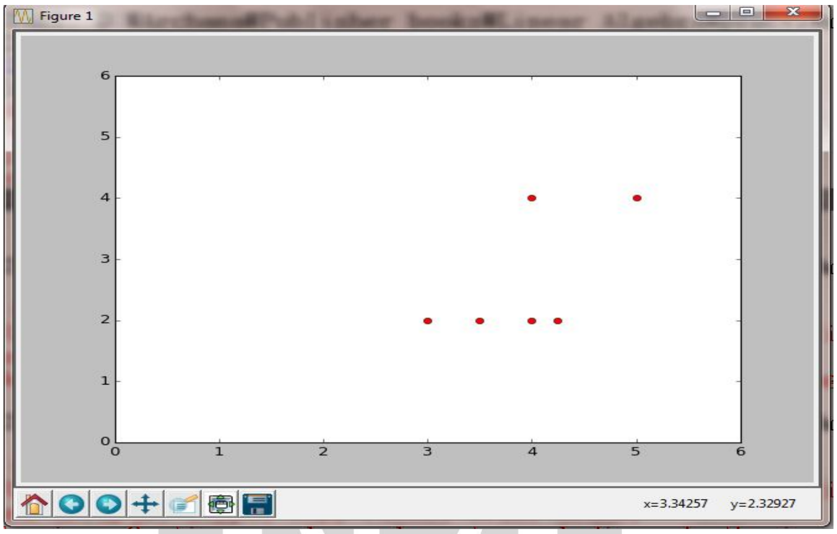


Fig. 1.2

Scaling

When complex numbers are multiplied with a positive real number changes the scale of coordinates.

For example when we halve each complex number in S: g(z) = 1/2 Z

This operation simply halves the real coordinate and the imaginary coordinate of each complex number. This will move all the points closer from the origin but also closer to each other this operation is called as scaling. The scale of the picture has changed.

Similarly, doubling each complex number moves the points farther from the origin and from each other.

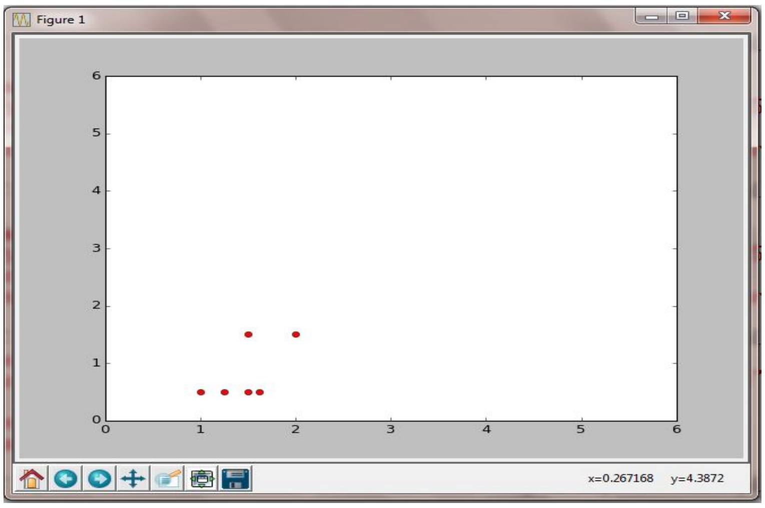


Fig. 1.3

Rotation

By 180^0: To rotate a complex number by 180^o, multiply z by -1.

E.g. if z = 2 + 4i, it will lie in first quadrant and if we wish to rotate it by 180^o. then multiply it by -1. ⇒ z = -2 - 4i which is the third quadrant.

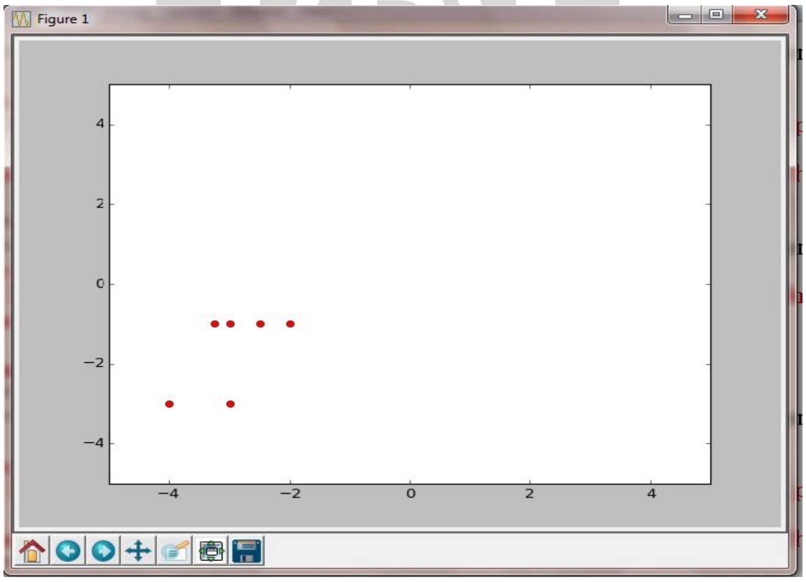


Fig. 1.4

By 90^0: To rotate a complex number by 90^o, multiply z by i.

e.g if z = x + iy then multiply z by i we get z_1 = x i + i (iy) = xi – y = -y + xi

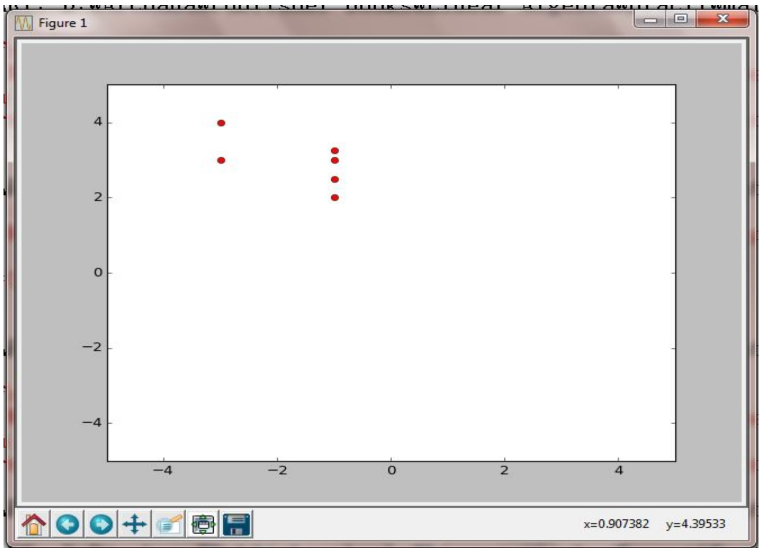


Fig. 1.5

Exercise 1.5

Plot the Graph Manually

(i) If S = {(2+ i), (2 + 4i), (4 + i), (4 + 4i) } then find the coordinates of the figure in other three quadrants also plot the graph

(ii) If S = { (1 + 2i), (2 + 3i), (3 + 3i)} then transform the fig by 2 + 2i

(iii) S1={(2 + 4i), (4 + 6i), (5 + 5i)}

1.6 PLOTTING OPERATIONS IN PYTHON

- **numpy.matplotlib:-** matplotlib is a plotting library for Python. It is used along with numpy to provide an environment that is an effective open source alternative for Mat Lab. It can also be used with graphics toolkits like PyQt and wx Python.
- **Matplotlib** module was first written by John D. Hunter. Since 2012, Michael Droettboom is the principal developer.
- **matplotlib.pyplot** is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

➢ matplotlib.pyplot.plot	:	Line graph
➢ matplotlib.pyplot.bar	:	Bar Graph
➢ matplotlib.pyplot.hist	:	Histogram
➢ matplotlib.pyplot.scatter	:	Scatter plot

We are going to use matplotlib.pyplot.plot() function for complex number plotting.

- **matplotlib.pyplot.plot() function :-** Plot lines and/or markers to the Axes.

  Syntax : matplotlib.pyplot.plot(*args, **kwargs)

  aArgs:- is a variable length argument, allowing for multiple x, y pairs with an optional format string.
  kwargs:- are line2D properties .

  For example:

  plot(x, y) *# plot x and y using default line style and color*

  plot(x, y, 'ro') *# plot x and y using red circle markers*

  The following format string characters are accepted to control the line style or marker:

Character	Description	Character	Description
'-'	solid line style	's'	square marker
'--'	dashed line style	'p'	pentagon marker
'-.'	dash-dot line style	'*'	star marker
':'	dotted line style	'h'	hexagon1 marker
'.'	point marker	'H'	hexagon2 marker
','	pixel marker	'+'	plus marker
'o'	circle marker	'x'	x marker
'v'	triangle_down marker	'D'	diamond marker
'^'	triangle_up marker	'd'	thin_diamond marker
'<'	triangle_left marker	'|'	vline marker
'>'	triangle_right marker	'_'	hline marker

The following color abbreviations are supported:

Character	Color	Character	Color
‘b’	Blue	‘m’	magenta
‘g’	Green	‘y’	yellow
‘r’	Red	‘k’	black
‘c’	Cyan	‘w’	white

- Line styles and colors are combined in a single format string, as in 'ro' for red circles.
- The *kwargs* can be used to set line properties such as label (for auto legends), line width, marker face color, etc.

  for example: plot([1,2,3], [1,2,3], 'ro-', label='line 1', line width=2)

  It prints red color lines with circle marker of width 2 pixel.
- Instead of writing abbreviations in format string these line properties can be set by keyword arguments.

  For example, you can set the color, marker, line style, line width, marker size and marker face color with:

  plot(x,y,color='blue', linestyle='solid', marker='o', markerfacecolor='red', markersize=12).

Plotting Set of Complex Numbers S in Python

```
import matplotlib.pyplot as plt
S={3 + 3j, 4 + 3j, 2 + 1j, 2.5 + 1j, 3+ 1j, 3.25 + 1j}
X = [x.real for x in S]
Y = [x.imag for x in S]
plt.plot(X,Y,'ro')
plt.axis([0, 5, 0, 5])
plt.show()
```

Output similar to fig.1.1

Using comprehension feature of Python we will create a new plot in which the points of S are translated, scaled and rotated

Translation:- Comprehension to create a new plot by adding complex no 1 + 2i to each element of S.

```
S1={x+1+2j for x in S}
```

Output fig1.2

Scaling:- Comprehension to create a new plot which points should be halves of the points in S.

```
S1={x/2 for x in S}
```

Output fig 1.3

Rotation by 180^0:- Comprehension to create a new plot by rotating all points in complex plane to 180^0

```
S1={x*-1 for x in S}
```

Output fig 1.4

Rotation by 90^0:- Comprehension to create a new plot by rotating all points in complex plane to 90^0

```
S1={x*1j for x in S}
```

Output fig 1.5

Exercise 1.6

Plot the Graph Using Matplotlib

(i) Create a new plot by using comprehension to rotate the points of S by 90 degrees, scaled by 3/4, and then shifted down by two units and to the right one unit.

(hint:Use a comprehension in which the points of S are multiplied by one complex number and added to another.

(ii) For above complex plane calculate the length of all points from origin.

(iii) Plot a complex plane to display a circle in lower right corner and rotate it to top left corner.

1.7 ABSTRACTING OVER THE FIELDS

In programming languages you have studied the concept of overloading. Overloading means using the same name for different procedures operating on values of different data types. For example in python when you use + operator for numeric data type it performs addition of 2 numbers, whereas when same operator is used for string data type it does concatenation of 2 strings.

```
>>10+5
15
>>'Acharya' + 'college'
Acharyacollege
```

Let us consider a procedure solve (a,b, c) to solve an equation ax + b = c where a is nonzero:

```
>>> def solve1(a,b,c): return (c-b)/a
>>> solve1(10, 5, 30)
2.5
```

This same procedure you can use to solve equations involving complex numbers.

For example to solve the equation (10 + 5i)x + 5 = 20:

```
>>> solve1(10+5j, 5, 20)
(1.2-0.6j)
```

- The procedure works even with complex arguments because the correctness of the procedure does not depend on what kind of arguments are supplied to it. It depends only on the fact that the division and multiplication are the inverse operator of each other and the subtract operator is the inverse of the add operator.
- Much of linear algebra concepts, theorems and procedures work not only for the real numbers, complex numbers but for other kinds of numbers as well.
- The concepts, theorems and procedures are stated in terms of the arithmetic operators +, -, * and /. Rely only on the basic laws:-

Commutativity	(a + b = b + a) and
Distributivity	(a(b + c) = ab + ac).

  So we can use any system of numbers, called a field.
- The generality of linear algebra is illustrated with three different fields.
  - R, the field of real numbers,
  - C, the field of complex numbers, and
  - GF(2), a field that consists of 0 and 1.

Solution of Equation with Complex Number

The procedure to solve equations with complex argument depends on the fact that the divide operator is the inverse of the multiply operator and the subtract operator is the inverse of the addition operator.

Example 1: Solve (10 + 5i) x + 5 = 20

Solution: (10 + 5i) x +5 = 20

Subtract 5 from both the sides

$(10 + 5i)\ x + 5\text{-}5 = 20\text{-}5$

$\therefore\quad (10 + 5\ i)\ x = 15$

$(2 + i)\ x = 3$

$\therefore\quad x = \frac{3}{(2+i)} \times \frac{2-i}{2-i} = \frac{6-3\ i}{4+1} = \frac{6-3i}{5}$

$\therefore\quad x = \frac{6}{5} - \frac{3}{5} i$ Ans

Example 2: If 2x -3i =3 – 9y i Find the value of x and y.

Solution: 2x -3i = 3 – 9y i

On equating real and imaginary part

2x = 3 and -3 = -9y

$\therefore\quad x = \frac{3}{2}$ and $y = \frac{1}{3}$ Ans

Exercise 1.7

Solve the Following:

(i) $x^2 = -13$ (ii) $x^2 = -19$ (iii) 4x - 8i = 7-4i (iv) $2x^2 + 27 = 0$ (v) $6x^2 - 2x + 3 = 0$

Answer:

(i) $i\sqrt{13}$ (ii) $i\sqrt{19}$ (iii) $\frac{7}{4} + i$ (iv) $\pm\frac{3i\sqrt{6}}{2}$ (v) $\frac{1 \pm i\sqrt{17}}{6}$

1.8 PLAYING WITH GF(2)

1.8.1 GF (2)

Galois Field of two elements 0 AND 1, named after Evariste Galois, also known as the smallest finite field. The two elements are nearly always called 0 and 1, being the additive and multiplicative identities, respectively.

The field's addition operation is given by the table below, which corresponds to the logical XOR operation. The field's multiplication operation corresponds to the logical AND operation.

Because GF(2) is a field, many of the familiar properties of number systems such as the rational numbers and real numbers are retained.

GF(2) = {0, 1}

Addition and Multiplication over GF(2) can be summarized in two small tables

+	0	1
0	0	1
1	1	0

X	0	1
0	0	0
1	0	1

It implies that addition is modulo 2. 1 + 1= 2

and z/p where p = 2 i.e. when the number 2 is divided by p it gives the remainder 0, similarly 1 + 0 = 1, when we divide 1 by 2 it gives remainder 1. Here negatives of 0 and 1 are again negative.

Multiplication in GF(2) is like an ordinary multiplication of 0 and 1.

Example: In GF(2) Field

2 = 1 + 1 = 0

3 = 2 + 1 = 0 + 1 = 1

4 = 3 + 1 = 1 + 1 = 2 = 0 etc.

Example: Multiplication In GF(2) Field

1. 0 + 1.1 + 0.1 +0.0 = 1
2. 1.1 + 1.1 +1.1 +1.1 = 0

Exercise 1.8.1

Solve the Following:

(i) $1 + 1 + 1 + 0$

(ii) $1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 1$

(iii) $(1 + 1 + 1) \cdot (1 + 1 + 1 + 1)$

(iv) $(1.1) + (1.0) + (1.1)$

(v) $(1+ 1 + 1 + 1 + 1)$

Answer:

(i) 1 (ii) 0 (iii) 0 (iv) 0 (v) 1

1.8.2 Applications of GF2

It is particularly useful in translating computer data as they are represented in binary forms. That is, computer data consist of combination of two numbers, 0 and 1, which are the components in Galois field whose number of elements is two. Representing data as a vector in a Galois Field allows mathematical operations to scramble data easily and effectively

- **Cryptography**

The most popular and widely used application of Galois Field is in Cryptography. Since each byte of data are represented as a vector in a finite field, encryption and decryption using mathematical arithmetic is very straight-forward and is easily manipulable.

The cryptosystem implemented by GF (2) achieves perfect secrecy.

For example A want to convey some secret message to B in encrypted format. They choose the key k uniformly from {♣,♥}. A has used the following encryption function to transform the plaintext bit p to a ciphertext bit c:

P	K	C
0	♣	0

0	♥	1
1	♣	1
1	♥	0

The encryption method is GF(2) addition .When we replace ♣ with 0 and ♥ with 1, the encryption table becomes the addition table for GF(2):

P	K	C
0	0	0
0	1	1
1	0	1
1	1	0

For each plaintext p ∈ GF(2), the function k→k + p (mappingGF(2) to GF(2)) is invertible.

Therefore, when the key k is chosen uniformly at random, the cipher text is also distributed uniformly. This shows that the scheme achieves perfect secrecy.

Instead of GF(2) if integers are used then there will be no uniform distribution

Encrypting long messages with G(2)

Suppose the message to be encrypted will consist of n bits. You should select an equally long sequence of key bits k1 . . . kn.

The plaintext p1 . . . pn is encrypted the cyphertext c1 . . . cn will be received one bit at a time:

c1 = k1 + p1

c2 = k2 + p2

...

cn = kn + pn

This system has perfect secrecy called as one-time pad in cryptography.

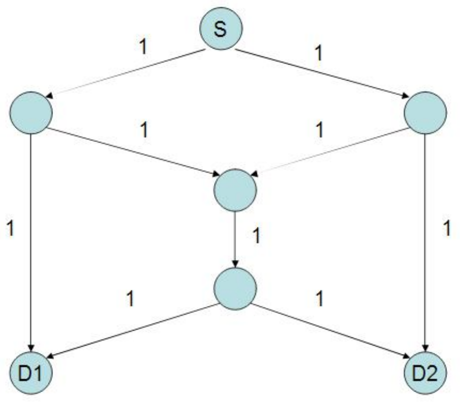
- **Network coding**

Linear network coding is a mathematical technique which may be used to improve a network's throughput, efficiency and scalability, as well as resilience to attacks and eavesdropping. Instead of simply relaying the packets of information they receive, the nodes of a network take several packets and combine them together for transmission. This may be used to attain the maximum possible information flow in a network

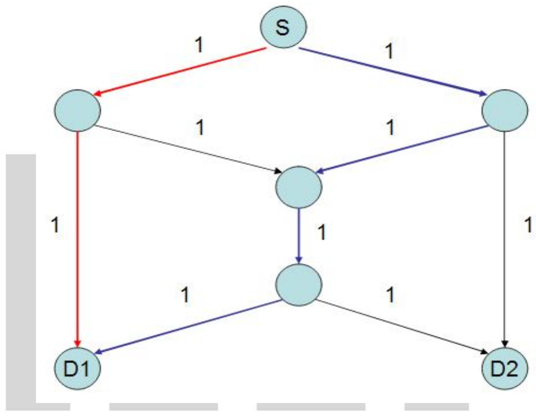
Traditionally, the way nodes moved data from source(s) to destination(s) is through packet "routing". Unfortunately, in some situations, simple packet routing does not allow you to achieve the actual capacity of network.

For example

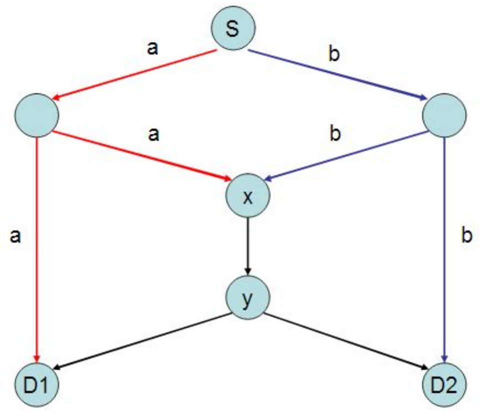
S wants to send 2 bits to both D1 and D2 and each link in the network has a capacity of 1 megabit per second.



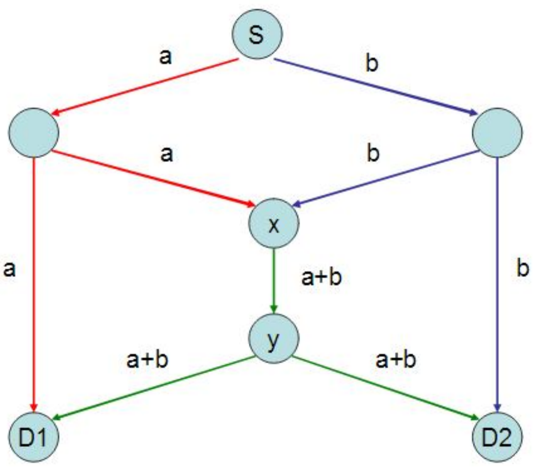
If communication is unicast means only in between s and d1 then there will be no problem



But if node s needs to stream bits to each of the two nodes d1 and d2 i.e multicast communication is impossible. We would need to get bits a and b to node y so that it can forward them to D2and D1, respectively



But if we use network coding rather than packet forwarding, we can XOR a and b and meet our 2 bit traffic requirement.



The XOR operation is nothing but a GF(2) addition is used to obtain a single bit. That single bit is transmitted as shown to the d1 and d2. D1 receives bit a and the sum a + b, so can also compute the bit b. D2 receives bit b and the sum a + b, so can also compute the bit a.