# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JNANA SANGAMA, BELGAVI-590018, KARNATAKA**

# A J INSTITUTE OF ENGINEERING & TECHNOLOGY

**(A unit of Laxmi Memorial Education Trust. (R))**
**NH - 66, Kottara Chowki, Kodical Cross - 575 006**

Offering Department

## DEPARTMENT OF MATHEMATICS

Beneficiary Department

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA)

# MASTER MANUAL

## Course: MATHEMATICS II FOR CSE STREAM
## Course Code: BMATS201

## II SEMESTER

Prepared By:

**Mrs. Vilma D'Souza**
Assistant Professor
Department of Mathematics

**Scheme: 2022**

**A J INSTITUTE OF ENGINEERING & TECHNOLOGY**

(A unit of Laxmi Memorial Education Trust. (R))
NH - 66, Kottara Chowki, Kodical Cross - 575 006

**Vision**

To produce top-quality engineers who are groomed for attaining excellence in their profession and competitive enough to help in the growth of nation and global society.

**Mission**

- To offer affordable high-quality graduate program in engineering with value education and make the students socially responsible.
- To support and enhance the institutional environment to attain research excellence in both faculty and students and to inspire them to push the boundaries of knowledge base.
- To identify the common areas of interest amongst the individuals for the effective industry- institute partnership in a sustainable way by systematically working together.
- To promote the entrepreneurial attitude and inculcate innovative ideas among the engineering professionals.

**Program Outcome**

**PO1**: Engineering knowledge Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

**PO2**: Problem Analysis Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

**PO3**: Design/development of solutions Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.

**PO4**: Conduct investigations of complex problems Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5**: Modern tool usage: create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

**PO6**: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7**: Environment sustainability Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8**: Ethics Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9**: Individual and team work Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10**: Communication communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions

**PO11**: Project management and finance Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12**: Lifelong learning recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

**Mathematics-II for CSE Stream**

**(Effective from the academic year 2022 -2023)**

**SEMESTER – II**

| | | | |
|---|---|---|---|
| **Course Code** | **BMATS201** | CIE Marks | **50** |
| **CREDITS** | **04** | SEE Marks | **50** |
| Teaching Hours/Week (L:T:P:S) | 2:2:2:0 | Exam Hours | **3 hours** |

**Course Type :** Integrated

**Total Hours of Pedagogy** 40 hours Theory + 10 to12 Lab

**Course objectives:** The goal of the course Mathematics-II for ECE stream (BMATE201) is

- **Familiarize** the importance of Integral calculus and Vector calculus.
- **Learn** vector spaces and linear transformations**.**
- **Develop** the knowledge of numerical methods and apply them to solve transcendental and differential equations.

**Module-1 Integral Calculus (8 hours)**

**Introduction to Integral Calculus in Computer Science &Engineering.**

**Multiple Integrals:** Evaluation of double and triple integrals, evaluation of double integrals by change of order of integration, changing into polar coordinates. Applications to find Area and Volume by double integral. Problems.

**Beta and Gamma functions:** Definitions, properties, relation between Beta and Gamma functions. Problems.

**Module-2: Vector Space and Linear Transformations (8 hours)**

**Introduction to Vector Calculus in Computer Science & Engineering.**

Scalar and vector fields. Gradient, directional derivative, curl and divergence – physical interpretation, solenoidal and irrotational vector fields. Problems.

**Curvilinear coordinates:** Scale factors, base vectors, Cylindrical polar coordinates, Spherical polar coordinates, transformation between cartesian and curvilinear systems, orthogonality. Problems

**Module-4: Numerical Methods -1 (8 hours)**

**Importance of numerical methods for discrete data in the field of computer science & engineering.**
Solution of algebraic and transcendental equations - Regula-Falsi and Newton-Raphson methods (only formulae). Problems.
Finite differences, Interpolation using Newton's forward and backward difference formulae, Newton's divided difference formula and Lagrange's interpolation formula (All formulae without proof). Problems.
**Numerical integration**: Trapezoidal, Simpson's $(1/3)_{rd}$ and $(3/8)_{th}$ rules (without proof). Problems.

**Module-5: Numerical Methods -2 (8 hours)**

**Introduction to various numerical techniques for handling Computer Science & Engineering applications.**
**Numerical Solution of Ordinary Differential Equations (ODE's):** Numerical solution of ordinary differential equations of first order and first degree – Taylor's series method, Modified Euler's method, Runge-Kutta method of fourth order and Milne's predictor-corrector formula (No derivations of formulae). Problems.

## Course outcomes :

At the end of the course the student will be able to :

**CO1:** Apply the concept of change of order of integration and variables to evaluate multiple integrals and their usage in computing area and volume

**CO2:** Understand the applications of vector calculus refer to solenoidal, and irrotational vectors. Orthogonal curvilinear coordinates

**CO3:** Demonstrate the idea of Linear dependence and independence of sets in the vector space, and linear transformation

**CO4** Apply the knowledge of numerical methods in modelling of various physical and engineering phenomena

**CO5:** Solve first order ordinary differential equations arising in engineering problems.

**CO6:** Get familiarize with modern mathematical tools namely MATHEMATICA/MATLAB/PYTHON/ SCILAB

**Weblinks and Video Lectures (e-Resources):**

1. http://nptel.ac.in/courses.php?disciplineID=111
2. http://www.class-central.com/subject/math(MOOCs)

3. http://academicearth.org/
4. VTU e-Shikshana Program
5. VTU EDUSAT Program

**Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50). The minimum passing mark for the SEE is 35% of the maximum marks (18 marks out of 50). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester end examination (SEE), and a minimum of 40% (40 marks out of 100) in the total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

**Continuous Internal Evaluation (CIE):**

The CIE marks for the theory component of the IC shall be 30 marks and for the laboratory component 20 Marks. CIE for the theory component of the IC

**CIE for the theory component of the IC :**

Three Tests each of 20 Marks; after the completion of the syllabus of 35-40%, 65-70%, and 90-100% respectively.

Two Assignments/two quizzes/ seminars/one field survey and report presentation/one-course project totalling 20 marks. Total Marks scored (test + assignments) out of 80 shall be scaled down to **30 marks.**

**CIE for the practical component of the IC**

- On completion of every experiment/program in the laboratory, the students shall be evaluated and marks shall be awarded on the same day. The 15 marks are for conducting the experiment and preparation of the laboratory record, the other 05 marks shall be for the test conducted at the end of the semester.

- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test (**duration 03 hours**) at the end of the 15th week of the semester/after completion of all the experiments (whichever is early) shall be conducted for 50 marks and scaled down to **05 marks**.

Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IC/IPCC for **20 marks**.

- The minimum marks to be secured in CIE to appear for SEE shall be 12 (40% of maximum marks) in the theory component and 08 (40% of maximum marks) in the practical component. The laboratory component of the IC/IPCC shall be for CIE only. However, in SEE, the questions from the laboratory component shall be included. The maximum of 05 questions is to be set from the practical component of IC/IPCC, the total marks of all questions should not be more than 25 marks. The theory component of the IC shall be for both CIE and SEE.

**Semester End Examination (SEE):**

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the subject (**duration 03 hours**)

- The question paper shall be set for 100 marks. The medium of the question paper shall be English/Kannada). The duration of SEE is 03 hours.
- The question paper will have 10 questions. Two questions per module. Each question is set for 20 marks. The students have to answer 5 full questions, selecting one full question from each module. The student has to answer for 100 marks and **marks scored out of 100 shall be proportionally reduced to 50 marks.**
- There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics under that module**

**Suggested Learning Resources:**

**Books (Title of the Book/Name of the author/Name of the publisher/Edition and Year)**

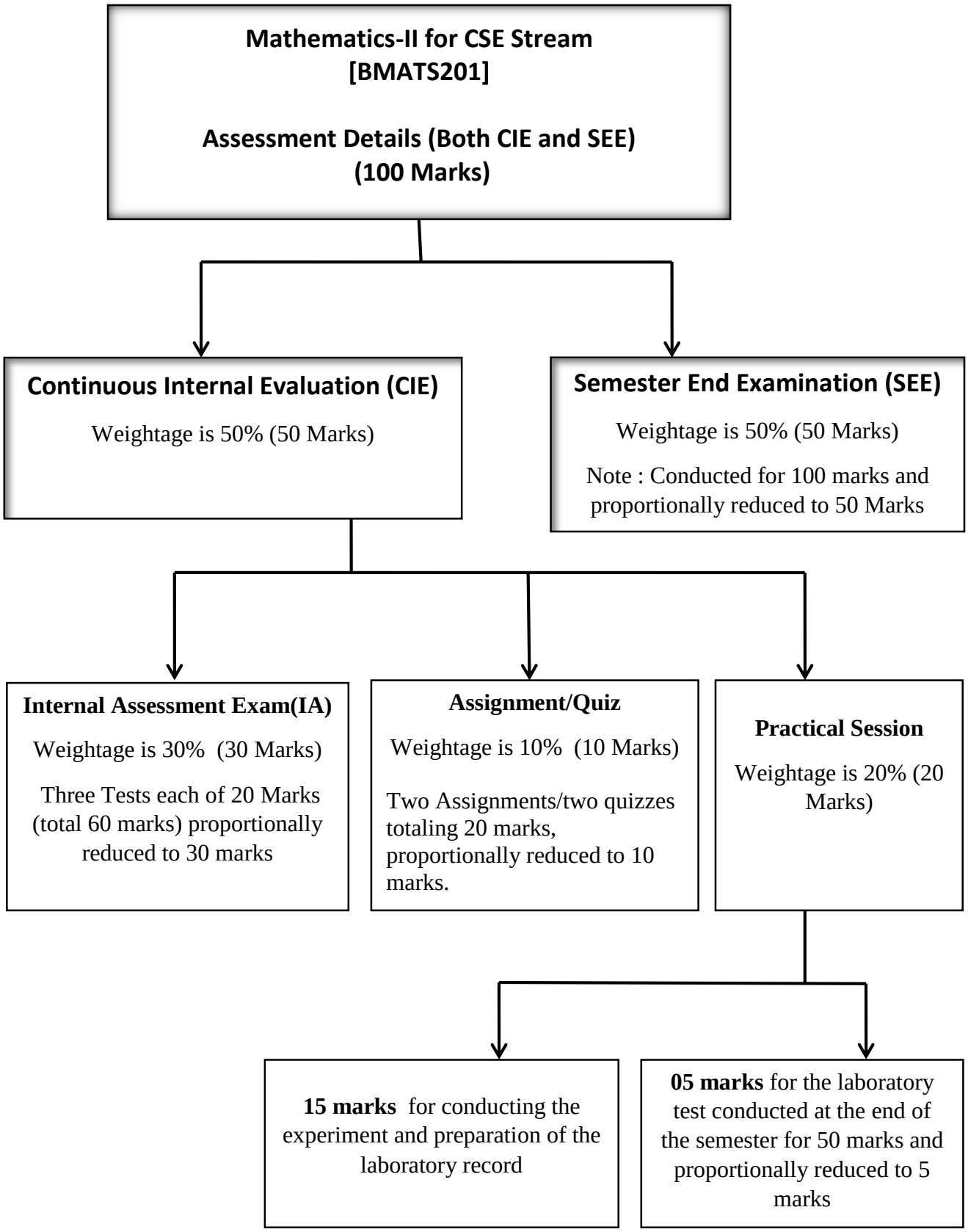
**Text Books :**


1. **B. S. Grewal**: "Higher Engineering Mathematics", Khanna Publishers, 44th Ed., 2021.
2. **E. Kreyszig**: "Advanced Engineering Mathematics", John Wiley & Sons, 10th Ed., 2018.
3. **David M Burton**: "Elementary Number Theory" Mc Graw Hill, 7th Ed.,2017.

**Reference Books :**


4. **V. Ramana**: "Higher Engineering Mathematics" McGraw-Hill Education, 11th Ed., 2017
5. **Srimanta Pal & Subodh C. Bhunia**: "Engineering Mathematics" Oxford University Press, 3 rd Ed., 2016.
6. **N.P Bali and Manish Goyal**: "A Textbook of Engineering Mathematics" Laxmi 26.10.2022 5 Publications, 10th Ed., 2022.
7. **C. Ray Wylie, Louis C. Barrett**: "Advanced Engineering Mathematics" McGraw – Hill Book Co., New York, 6th Ed., 2017.
8. **Gupta C.B, Sing S.R and Mukesh Kumar**: "Engineering Mathematic for Semester I and II", Mc-Graw Hill Education(India) Pvt. Ltd 2015.
9. **H. K. Dass and Er. Rajnish Verma**: "Higher Engineering Mathematics" S. Chand Publication, 3rd Ed., 2014.
10. **James Stewart**: "Calculus" Cengage Publications, 7th Ed., 2019.
11. **David C Lay**: "Linear Algebra and its Applications", Pearson Publishers, 4th Ed., 2018.
12. **Gareth Williams:** "Linear Algebra with Applications", Jones Bartlett Publishers Inc., 6th Ed., 2017.
13. **Gilbert Strang:** "Linear Algebra and its Applications", Cengage Publications, 4th Ed. 2022.

**Assessment Process for Mathematics-II for CSE Stream**



## List of Laboratory experiments (2 hours/week per batch/ batch strength 15) 10 lab sessions + 1 repetition class + 1 Lab Assessment

1) Program to compute area, surface area, volume and centre of gravity
2) Evaluation of improper integrals
3) Finding gradient, divergent, curl and their geometrical interpretation
4) Computation of basis and dimension for a vector space and Graphical representation of linear transformation
5) Computing the inner product and orthogonality
6) Solution of algebraic and transcendental equations by Ramanujan's, Regula-Falsi and Newton-Raphson method
7) Interpolation/Extrapolation using Newton's forward and backward difference formula
8) Computation of area under the curve using Trapezoidal, Simpson's $(1/3)_{rd}$ and $(3/8)_{th}$ rule
9) Solution of ODE of first order and first degree by Taylor's series and Modified Euler's method
10) Solution of ODE of first order and first degree by Runge-Kutta $4_{th}$ order and Milne's predictor-corrector method

**Instructions to students**

- Students should report to the concerned labs as per the given timetable. Be regular to the Lab Do not come late to the Lab
- Students should make an entry in the log book whenever they enter the labs during practical.
- Students must bring Observation book, record and manual along with pen, pencil, and eraser Etc., no borrowing from others.
- Students must use the computers carefully, as they are expensive. Each person may only use one computer at a time
- Any damage to the lab computers will be viewed seriously.
- Students may not install software on lab computers. If you have a question regarding specific software that you need to use, contact the concerned Faculty and Labs support team.
- When the experiment is completed, students should shut down the computers and make the counter entry in the logbook
- Wear your College ID card
- Avoid unnecessary talking while doing the experiment
- Do not panic if you do not get the output
- Students should not leave the lab without concerned faculty's permission.
- Before leaving the lab, students should check whether they have switched off the computers and the power supplies and kept their chairs properly.

## Rules for Maintaining Record

- Write your name, USN and name of the subject on the outside front cover of the record. Write the same information in the first page inside the cover page.
- Update Table of Contents every time you start each new experiment or topic.
- Always use pen and write neatly and clearly
- Start each new topic (experiment, notes, calculation, etc.) on a right-side (odd numbered) page
- Obvious care should be taken to make it readable, even if you have bad handwriting
- Date to be written on every page on the top right-side corner
- On each right-side page
  - Title of experiment
  - Aim/Objectives
  - Theory
  - Procedure described clearly in steps
  - Result
- On each left side page
  - Diagrams
  - Tables
  - Graphs
- Use labels and captions for figures and tables
- Attach printouts and plots of data as needed. Stick printouts (A4 Size) on the right side of the lab record
- Strictly observe the instructions given by the Teacher/ Lab Instructor.

**Introduction**

**WHAT IS PYTHON?**

Python is a computer programming language. It is a high-level general purpose scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Beginner-level language that supports development of wide range of applications.

## History of Python

- Python was developed by ***Guido van Rossum*** in the ***late eighties*** and ***early nineties*** at the ***National Research Institute*** for Mathematics and Computer Science in the ***Netherlands***. Guido van Rossum (1987) named it after the BBC television show 'Monty Python's Flying Circus.' (Comedy show).
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**PYTHON USED FOR...**

**Python is used in many places.**

**Web and Internet Development**

**Desktop GUI Applications**

**Scientific and Numeric**

**Software Development**

**Education**

**Business Applications**

**Games and 3D Graphics**

**Network programming**

**Database Access**

**One of the recent growing field of expertise is 'data science'. Many data scientists use**

**Python for their day-to-day work.**

**PYTHON'S MAJOR FEATURES**

**Python Features**

**It is easy to read and write.**

**Python is an interpreted, interactive, object-oriented programming language**

**It incorporates modules, exceptions, dynamic typing, very high level dynamic data**

**types, and classes.**

**It has interfaces to many system calls and libraries, as well as to various window**

**systems.**

**It is used as an extension language for applications that need a programming interface.**

- Python's features include −
- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- Python is portable: it runs on many Unix variants including Linux and macOS, and on Windows.
- Python includes a comprehensive base library.

INTRODUCTION TO ANACONDA

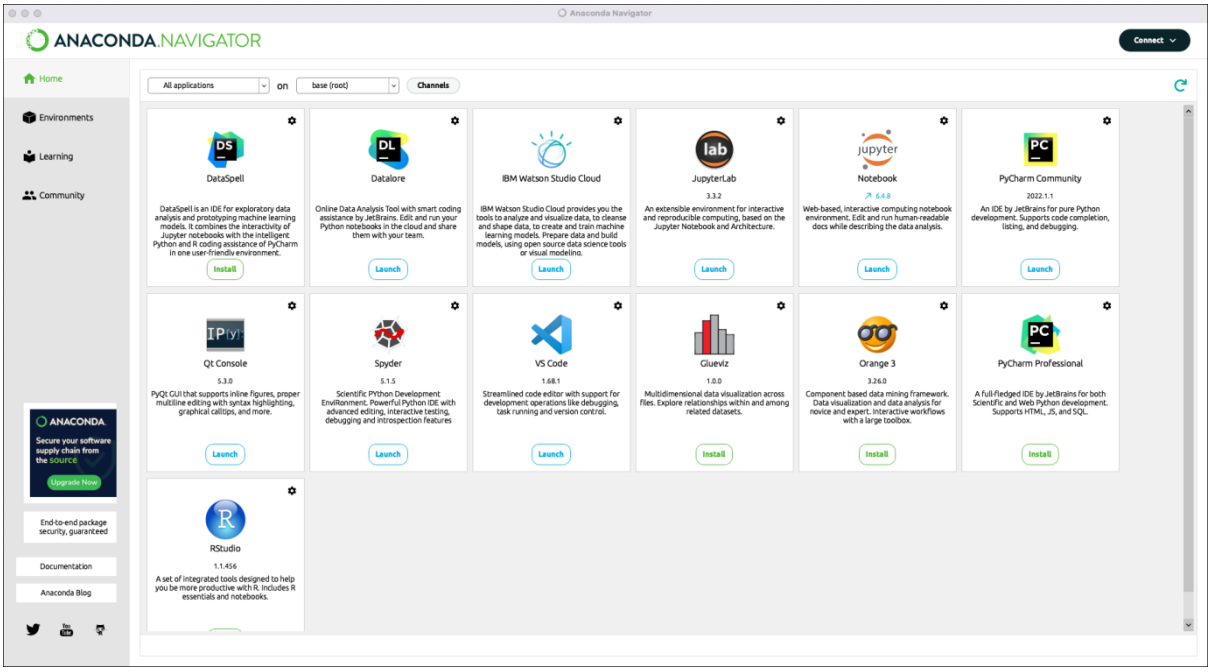
Anaconda is a package manager, environment manager, and Python distribution with a collection of 1,500+ open source packages with free community support. Anaconda is free and easy to install and can be used on Windows, macOS, or Linux.

Anaconda can be downloaded from

https://www.anaconda.com/products/individual

After installing Anaconda, we use Anaconda Navigator to launch applications and easily manage packages, environments and channels without using command-line commands.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in the terminal window.



**APPLICATIONS IN NAVIGATOR**

**The following applications are available by default in Navigator:**

**JupyterLab**
**Jupyter Notebook**
**Spyder**
**PyCharm**
**VSCode**
**Glueviz**
**Orange 3 App**
**RStudio**
**Anaconda Prompt (Windows only)**
**Anaconda PowerShell (Windows only)**

**JUPYTER NOTEBOOK**

**The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communi-**
**cating the results.**

**KEYWORDS AND IDENTIFIERS Keywords are the reserved words in Python used by Python interpreter to recognize**
**the structure of the program.**
**Identifiers are the name given to entities like class, functions, variables etc. It helps to differentiate one entity from another. Identifiers can be a combination of letters from "a" to "z", from "A" to "Z" and digits from**
**"0" to "9" or special character _.**

**PYTHON IDENTIFIERS**

- A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.
- Keywords cannot be used as identifiers.
- Only special character used in identifiers is _.
- Identifier can be of any length.

**Reserved Words**

- The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

| | | |
|---|---|---|
| and | exec | Not |
| assert | finally | or |
| break | for | pass |
| class | from | print |
| continue | global | raise |
| def | If | return |
| del | import | try |
| elif | in | while |
| else | is | with |
| except | lambda | yield |
| False | True | None |

**DATA TYPES**

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

**Text Type str**

**Numeric Types int, float, complex**

**Sequence Types list, tuple, range**

**Mapping Type dict**

**Set Types set, frozenset**

**Boolean Type bool**

**Binary Types bytes, bytearray, memoryview**

## QUOTATION IN PYTHON

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

## COMMENTS IN PYTHON

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

```
# First comment

print ("Hello, Python!") # second comment
```

This produces the following result −

```
Hello, Python!
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

### Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

## PYTHON - VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

### Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example −

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
print (counter)
print (miles)
print (name)
```

Here, 100, 1000.0 and "John" are the values assigned to *counter*, *miles*, and *name* variables, respectively. This produces the following result −

```
100
1000.0
John
```

**Multiple Assignment**

Python allows you to assign a single value to several variables simultaneously. For example

```
a = b = c = 1
```

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables. For example −

```
a,b,c = 1,2,"john"
```

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

EXAMPLES:

a=2.4

x=3

y = ’a ’

z=" Hello "

p r i n t ( type ( a ) ) #type ( ) gives the data type of the v a r i a b l e .

p r i n t ( type ( x ) )

p r i n t ( type ( y ) )

p r i n t ( type ( z ) )

EXAMPLE FOR NUMERIC TYPES:

x = 1 # i n t

y = 2.8 # f l o a t

z = 1 j # complex

X = 35e3 # e i n d i c a t e s the power of 10

Y = 12E4

Z = −87.7e100

EXAMPLE FOR TEXT TYPE:

OPERATORS

Operators are the constructs which can manipulate the value of operands.

Consider the expression 4 + 5 = 9. Here, 4 and 5 are called operands and + is called operator.

**Types of Operator**

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

All the common algebraic operators presented in the following table are available in Python.

Addition +

Subtraction −

Multiplication ∗

Division /

Integer Division //

Power **

Remainder %

Comparison operators:

< less than

> greater than

<= less than or equal to

>= greater than or equal to

== equal to

! = not equal to

Logical operators:

and logical and

or logical or

not logical not

ASSIGNMENT OPERATOR

=, is used as assignment operator. For example x = 5, means 5 is stored to variable

x.

+ =. The expression a+ = 3, give the result of a = a + 3.

− =. The expression x− = 3, give the result of x = x − 3.

∗ =. The expression a∗ = 3, give the result of a = a ∗ 3.

/ =. The expression a/ = 3, give the result of a = a/3.

// =. The expression a// = 3, give the result of a = a//3.

% =. The expression a% = 3, give the result of a = a%3.

∗∗ =. The expression a∗ = 3, give the result of a = a ∗ ∗3.

IDENTITY OPERATORS

is  Returns True if both variables are the same object

is not  Returns True if both variables are not the same object

MEMBERSHIP OPERATORS

in Returns True if a sequence with the specified value is present in the object

not in Returns True if a sequence with the

specified value is not present in the

object

## PYTHON LISTS

Python  Lists are used to group related data together. They are collection of data items. Lists are used to store multiple items in a single variable. We enclose the items with in square brackets[ ]and separate them by commas.
Lists are created using square brackets: for example
a=[1,2,3]

Ex:
```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

# PYTHON TUPLES

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values.

**Ex:**

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

Tuples are created using round brackets: for example

a=(1,2,3)

items are ordered, allow duplicate values.

List items are changeable, whereas; Tuple items are unchangeable.

INPUT STATEMENT

The input() function allows user input.

SYNTAX:

input(prompt)

prompt– A String, representing a default message before the input.

For example:

x = in p ut ( ’ Enter your name : ’ )

p r i n t ( ’ Hello , ’ +x )

p r i n t ( ’ Hello , ’ , x )

p r i n t ( type ( x ) )

To read integer then the following to be used.

x = i n t ( in p ut ( ’ Enter an i n t e g e r : ’ ) )

p r i n t ( type ( x ) )

To read float then the following to be used.

x = f l o a t ( in p ut ( ’ Enter an f l o a t : ’ ) )

p r i n t ( type ( x ) )

OUTPUT STATEMENTS

Python provides the print() function to display output to the standard output devices.

SYNTAX:

print(value(s),sep= ’ ’, end = ‘\n’, file=file, flush=flush)

value(s) – Any value, and as many as you like. Will be converted to string before printed

sep=‘separator’ – (Optional) Specify how to separate the objects, if there is more than

one. Default :’ ’

end=‘end’ – (Optional) Specify what to print at the end.Default : ‘\n’

file – (Optional) An object with a write method. Default :sys.stdout

flush – (Optional) A Boolean, specifying if the output is flushed (True) or buffered (False).

Default: False

# Basics of Python

https://drive.google.com/file/d/1gVG2IJ8BIjhYDwDx6jWJns59h9dGOGVi/view?usp=share_link

Programming Structures

Conditional structure

What is conditioning in Python?

• Based on certain conditions, the flow of execution of the program is determined using proper syntax.

How to use if conditions?

- if statement — for implementing one-way branching
- if-else statements —for implementing two-way branching
- nested if statements —for implementing multiple branching
- if-elif ladder — for implementing multiple branching

```
# Syntax:
if condition:
statements
```

Example: #Check if the given number is positive

```
a=int (input (" Enter an integer: "))
if a>0:
print (" Entered value is positive ")
```

```
# Syntax:
# if condition 1:
# Statements 1
# elif condition 2:
# Statements 2
# elif condition 3:
# Statements 3
# else :
# Statements 4
# If condition 1 is True - Statements 1 will be executed.
# else if condition 2 is True - Statements 2 will be executed and so on
# If any of the conditions is not True then statements in else block is
executed .
```

# LAB 1: Program to compute area, surface area, volume and centre of gravity

**1.1 Objectives:**
Use python
1. to evaluate double integration.
2. to compute area and volume.
3. to calculate center of gravity of 2D object.

**Syntax for the commands used:**
1. Data pretty printer in Python:
   pprint ()
2. integrate:
   integrate ( function ,( variable , min_limit , max_limit ))

1. Evaluate the integral $\int_0^1 \int_0^x (x^2 + y^2) dydx$

```
from sympy import *
x,y,z= symbols ('x y z')
w1= integrate (x ** 2+y ** 2 ,(y,0,x) ,(x,0,1))
print (w1)
```

Output

```
1/3
```

2. Evaluate the integral $\int_0^3 \int_0^{3-x} \int_0^{3-x-y} xyz\, dzdydx$

```

```

Output

```
81/80
```

3. Prove that $\int\int(x^2+y^2)dxdy = \int\int(y^2+x^2)dxdy$

```
from sympy import *
x= symbols('x y z')
w3= integrate (x ** 2+y ** 2,y,x)
display (w3)
w4= integrate (x ** 2+y ** 2,x,y)
display (w4)
```

Output

$$\frac{x^3\cdot y}{3} + \frac{x\cdot y^3}{3}$$
$$\frac{x^3\cdot y}{3} + \frac{x\cdot y^3}{3}$$

## 1.3 Area and Volume

Area of the region R in the cartesian form is $\int\int dxdy$

4. Find the area of an ellipse by double integration. $A = 4\int_0^a \int_0^{\frac{b}{a}}$

```
from sympy import *
x= symbols ('x y a b')
a=4
b=6
w3=4* integrate (1 ,(y,0 ,(b/a)* sqrt (a ** 2-x ** 2)) ,(x,0,a))
print (w3)
```

Output

24.0 pi

**1.4 Area of the region R in the polar form is** $\int\int r dr d\theta$

5. Find the area of the cardioid r = a (1 + cosθ) by double integration.

```
from sympy import *
r= symbols ('r t a')
#a=4
w3=2* integrate (r, (r,0,a*(1+cos (t))) , (t,0,pi))
pprint (w3)
```

Output

$$\frac{3\pi a^2}{2}$$

**1.5 Volume of a solid is given by** $\int_V \int\int dx\, dy\, dz$

6. Find the volume of the tetrahedron bounded by the planes x=0, y=0 and z=0, $\frac{x}{a}+\frac{y}{b}+\frac{z}{c}=1$

```
from sympy import *
x= symbol ('x y z')
a= Symbol ('a b c')
w2= integrate (1 ,(z,0,c*(1-x/a-y/b)) ,(y,0,b*(1-x/a)) ,(x,0,a))
print (w2)
```

Output

$$a * b * c/6$$

# LAB 2: Evaluation of improper integrals, Beta and Gamma functions

## 2.1 Objectives:

Use python

1. to evaluate improper integrals using Beta function.
2. to evaluate improper integrals using Gamma function.

### Syntax for the commands used:

1. gamma

```
math.gamma(x)
```

Parameters :

x : The number whose gamma value needs to be computed.

2. beta

```
math.beta(x,y)
```

Parameters :

x ,y: The numbers whose beta value needs to be computed.

3. **Note:** We can evaluate improper integral involving infinity by using inf.

1.Evaluate $\int_0^\infty e^{-x}dx$

```
from sympy import *
x= Symbol ('x')
w1= integrate (exp (-x) ,(x,0, float ('inf ')))
print ( simplify (w1))
```

Output

1

**Gamma function is** $\Gamma(n) = \int_0^\infty e^{-x}x^{n-1}dx$

2. Evaluate Γ(5) by using definition

```
from sympy import *
x= Symbol ('x')
w1= integrate (exp (-x)*x ** 4 ,(x,0, float ('inf ')))
print ( simplify (w1))
```

Output:

24

3. Evaluate $\int_0^{\infty} e^{-st} cos(4t) dt$

```
from sympy import *
t,s= symbols ('t,s')
# for infinity in sympy we use oo
w1= integrate (exp (-s*t)*cos (4*t) ,(t,0,oo))
display ( simplify (w1))
```

Output

$$\begin{cases} \frac{s}{s^2+16} & \text{for } |\arg(s)| < \frac{\pi}{2} \\ \int_0^{\infty} e^{-st} \cos(4t)\, dt & \text{otherwise} \end{cases}$$

4. Find Beta(3,5), Gamma(5)

```
from sympy import beta , gamma
m= float(input ('m :'))
n= float(input ('n :'))
s= beta (m,n)
t= gamma (n)
print ('gamma (',n,') is', round(t,3))
print ('Beta (',m, n,') is', round(s,3))
```

**Output**

m :3
n :5

gamma ( 5.0 ) is 24.000
Beta ( 3.0, 5.0 ) is 0.010

5. Calculate Beta(5/2,7/2) and Gamma(7/2).

```
from sympy import beta , gamma
m= float ( input ('m : '))
n= float ( input ('n :'))
s= beta (m,n)
t= gamma (n)
print ('gamma (',n,') is', round(t,3))
print ('Beta (',m, n,') is', round(s,3))
```

**Output**

m : 2.5
n :3.5
gamma ( 3.5 ) is 3.323
Beta ( 2.5 3.5 ) is 0.037

6. Verify that Beta(m, n) = [Gamma(m) Gamma(n)] / Gamma(m + n)
   for m=5 and n=7

```
from sympy import beta , gamma
m=5
n=7
m= float (m)
n= float (n)
s= beta (m,n)
t=( gamma (m)* gamma (n))/ gamma (m+n)
print (s,t)
if (abs (s-t)<=0. 00001 ):
        print ('beta and gamma are related ')
else :
        print ('beta and gamma are not related')
```

**Output**

0.000432900432900433 0.000432900432900433
beta and gamma are related

## LAB 3: Finding gradient, divergence, curl and their geometrical interpretation

1.1 Objectives:
Use python
1.Finding gradient
2. Finding divergent
3. Finding curl

1) Find gradient of $F = x^2yz$

```
#To find gradient of a scalar point function x^2yz
from sympy.physics.vector import *
from sympy import var
var('x,y,z')
v=ReferenceFrame('v')
F=v[0]**2*v[1]*v[2]
G=gradient(F,v)
F=F.subs([(v[0],x),(v[1],y),(v[2],z)])
print("Given scalar function F=")
display(F)
G=G.subs([(v[0],x),(v[1],y),(v[2],z)])
print("\n Gradient of F=")
display(G)
```

**Output:** Given scalar function $F = x^2yz$

Gradient of F=$2xyzv_x + xzv_y + x^2yv_z$

2) Find divergence of F=x^2yi+yz^2j+x^2zk

```
#To find divergence of F=x^2yi+yz^2j+x^2zk
from sympy.physics.vector import *
from sympy import var
var('x,y,z')
v=ReferenceFrame('v')
F=v[0]**2*v[1]*v.x+v[1]*v[2]**2*v.y+v[0]**2*v[2]*v.z
G=divergence(F,v)
F=F.subs([(v[0],x),(v[1],y),(v[2],z)])
print("Given vector point function is ")
display(F)
G=G.subs([(v[0],x),(v[1],y),(v[2],z)])
print("Divergence of F=")
display(G)
```

**Output:**
Given vector function F=x^2yi+yz^2j+x^2zk

Divergence of F=$x^2 + 2xy + z^2$

3) Find curl of $\vec{F} = xy^2\hat{\imath} + 2x^2yz\hat{\jmath} - 3yz^2\hat{k}$

```
#To find curl of F = xy^2 i + 2x^2yz j - 3yz^2 k
from sympy.physics.vector import *
from sympy import var
var('x,y,z')
v=ReferenceFrame('v')
F=v[0]*v[1]**2*v.x+2*v[0]**2*v[1]*v[2]*v.y-3*v[1]*v[2]**2*v.z
G=curl(F,v)
F=F.subs([(v[0],x),(v[1],y),(v[2],z)])
print("Given vector point function is ")
display(F)
G=G.subs([(v[0],x),(v[1],y),(v[2],z)])
print("curl of F=")
display(G)
```

**Output:** Given vector function $\vec{F} = xy^2\hat{\imath} + 2x^2yz\hat{\jmath} - 3yz^2\hat{k}$

Curl of F= $(-2x^2y - 3z^2)\hat{v}_x + (4xyz - 2xy)\hat{v}_z$

## LAB-4: Computation of basis and dimension for a vector space and graphical representation of linear transformation

Objectives:

Use python:

1. to verify the Rank nullity theorem of given linear transformation
2. to compute the dimension of vector space
3. to represent linear transformations Graphically

**Rank Nullity Theorem**

Verify the rank-nullity theorem for the linear transformation $T : R^3 \rightarrow R^3$ defined by T(x, y, z) = (x + 4y + 7z, 2x + 5y + 8z, 3x + 6y + 9z)

```
import numpy as np
from scipy.linalg import null_space
# Define a linear transformation interms of matrix
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# Find the rank of the matrix A
rank = np.linalg.matrix_rank(A)
print("Rank of the matrix",rank)
# Find the null space of the matrix A
ns = null_space(A)
print("Null space of the matrix",ns)
# Find the dimension of the null space
nullity = ns.shape[1]
print("Null space of the matrix",nullity)
# Verify the rank-nullity theorem
if rank + nullity == A.shape[1]:
    print("Rank-nullity theorem holds.")
else:
    print("Rank-nullity theorem does not hold.")
```

**Output:** Rank of the matrix 2

Null space of the matrix [[-0.40824829] [ 0.81649658] [-0.40824829]]

Null space of the matrix 1 Rank-nullity theorem holds.

**Dimension of Vector Space**

1) Find the dimension of subspace spanned by the vectors (1,2,3) ,(2,3,1) and (3,1,2).

```
import numpy as np
# Define the vector space V
V = np.array([ [1, 2, 3], [2, 3, 1], [3, 1, 2]])
# Find the dimension and basis of V
basis = np.linalg.matrix_rank(V)
dimension = V.shape[0]
```

```
print("Basis of the matrix",basis)
print("Dimension of the matrix",dimension)
```

**Output:** Basis of the matrix 3
Dimension of the matrix 3

Extract the linearly independent rows in given matrix

```
import numpy as np
array = np.array([[ 1, 1, 0], [ 0, 0, 0], [ 0, 2, 3], [ 0, 0, 0], [ 0, -1, 1], [ 0, 0, 0]])
print("Original array:")
print(array)
temp = {(0, 0, 0)}
result = []
for idx, row in enumerate(map(tuple, arra)):
    if row not in temp:
        result.append(idx)
print("\nNon-zero unique rows:")
print(arra[result])
```

**Output:** Original array: [[ 1 1 0] [ 0 0 0] [ 0 2 3] [ 0 0 0] [ 0 -1 1] [ 0 0 0]]
Non-zero unique rows: [[ 1 1 0] [ 0 2 3] [ 0 -1 1]]

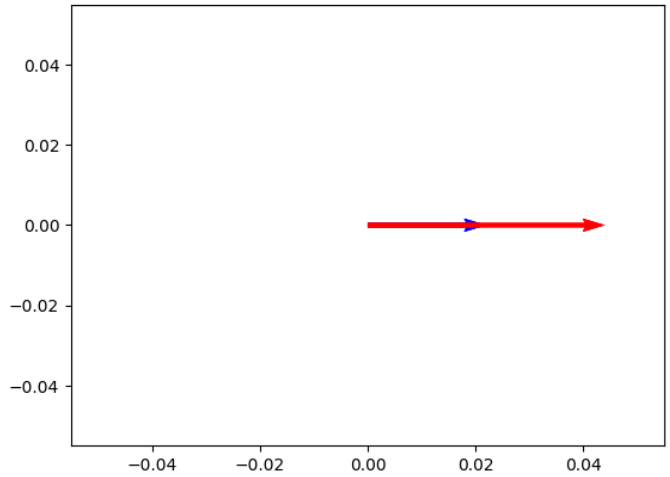
## Horizontal stretch:

Represent the horizontal stretch transformation geometrically T:$R^2 \rightarrow R^2$ geometrically

Find the image of the vector (10,0) when it is stretched horizontally by 2 units

```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[10 ,0]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[2,0],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =50)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =50)
plt . show ()
```

**Output:**



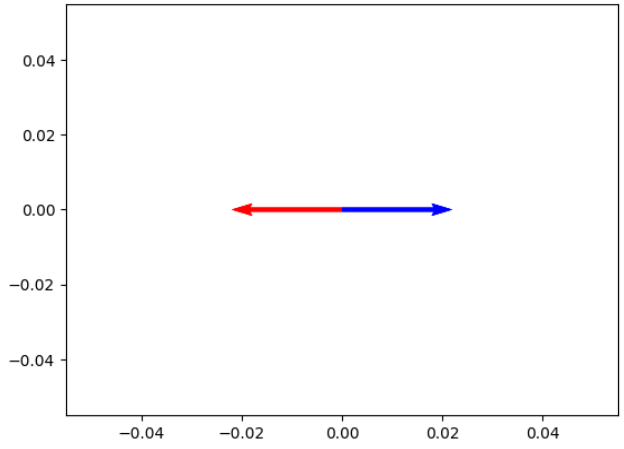

## Reflection:

Represent the reflection transformation T : $R^2 \rightarrow R^2$ geometrically.

Find the image of vector (10, 0) when it is reflected about y axis.

```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[10 ,0]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[-1,0],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =50)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =50)
plt . show ()
```

**Output:**




## Rotation:

Represent the rotation transformation T: $R^2 \rightarrow R^2$ geometrically.

Find the image of vector (10, 0) when it is rotated by π/2 radians.

```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[10 ,0]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[0,-1],[1,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =50)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =50)
plt . show ()
```

**Output:**

## Shear Transformation

Represent the Shear transformation T: $R^2 \rightarrow R^2$ geometrically.

Find the image of (2, 3) under shear transformation.

```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[2,3]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[1,2],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
print (" Image of given vectors is:", V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =20)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =20)
plt . show ()
```

**Output:**

## Composition

Represent the composition of two 2D transformations.

Find the image of vector (10, 0) when it is rotated by π/2 radians then stretched horizontally 2 units.

```
import numpy as np
```

```
import matplotlib . pyplot as plt
V = np. array ([[2,3]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[0,-1],[1,0]])
B=np. matrix ([[2,0],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V3=B*V2
V2=np. array (V2)
V3=np. array (V3)
print (" Image of given vectors is:", V3)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =20)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =20)
plt . quiver (*origin , V3[0,:], V3[1,:], color =['g'], scale =20)
plt . title ('Blue = original , Red = Rotated ,Green = Rotated + Streached ')
plt . show ()
```

**Output:**

**Exercise:**

1. Verify the rank nullity theorem for the following linear transformation

a) T: $R^2 \rightarrow R^3$ defined by T(x, y) = (x + 4y, 2x + 5y, 3x + 6y).

Ans: Rank=2, Nullity=1, RNT verified

b) T: $R^3 \rightarrow R^4$ defined by T(x, y, z) = (x+4y−z, 2x+5y+8z, 3x+y+2z, x+y+z).

Ans: Rank=3, Nullity=1, RNT verified

2. Find the dimension of the subspace spanned following set of vectors

a) S = (1, 2, 3, 4), (2, 4, 6, 8), (1, 1, 1, 1)

Ans: Dimension of subspace is 2

b) S = (1,−1, 3, 4), (2, 1, 6, 8), (1, 1, 1, 1), (3, 3, 3, 3)

Ans: Dimension of subspace is 3

3. Find the image of (1, 3) under following 2D transformations

a) Horizontal stretch

b) Reflection

c) Shear

d) Rotation

## LAB 5: Computing the inner product and orthogonality

5.1 Objectives:
Use python
1. to compute the inner product of two vectors.
2. to check whether the given vectors are orthogonal.

5.2 Inner Product of two vectors

1. Find the inner product of the vectors (2, 1, 5, 4) and (3, 4, 7, 8).

```
import numpy as np
# initialize arrays
A = np. array ([2, 1, 5, 4])
B = np. array ([3, 4, 7, 8])
#dot product
output = np. dot(A, B)
print ( output )
```

Output : 77

## 5.3 Checking orthogonality

Verify whether the following vectors (2, 1, 5, 4) and (3, 4, 7, 8) are orthogonal.

```
import numpy as np
# initialize arrays
A = np. array ([2, 1, 5, 4])
B = np. array ([3, 4, 7, 8])
#dot product
output = np. dot(A, B)
print ('Inner product is :',output )
if output ==0:
print ('given vectors are orthognal ')
else :
print ('given vectors are not orthognal ')
```

Output: Inner product is : 77
given vectors are not orthognal

5.4 Exercise:
1. Find the inner product of (1, 2, 3) and (3, 4, 5).
Ans: 26
2. Find the inner product of (1,−1, 2, 1) and (4, 2, 1, 0).

Ans: 4
3. Check whether the following vectors are orthogonal or not
a) (1, 1,−1) and (2, 3, 5). Ans: True
b) (1, 0, 2, 0) and (4, 2,−2, 5). Ans: True
c) (1, 2, 3, 4) and (2, 3, 4, 5) . Ans: False

## LAB 6: Solution of algebraic and transcendental equation by Regula-Falsi and Newton-Raphson method

6.1 Objectives:

Use python
1. to solve algebraic and transcendental equation by Regula-Falsi method.
2. to solve algebraic and transcendental equation by Newton-Raphson method.

6.2 Regula-Falsi method to solve a transcendental equation

Obtain a root of the equation x3 − 2x − 5 = 0 between 2 and 3 by regula-falsi method. Perform 5 iterations.

```
from sympy import *
x= Symbol ('x')
g = input ('Enter the function ')
f= lambdify (x,g)
a= float ( input ('Enter a values :'))
b= float ( input ('Enter b values :'))
N=int( input ('Enter number of iterations :'))
for i in range (1,N+1):
        c=(a*f(b)-b*f(a))/(f(b)-f(a))
        if ((f(a)*f(c)<0)):
                b=c
        else :
                a=c
                print ('iteration %d \t the root %0.3f \t function value %0.3f \n'%(i,c,f(c)))
```

## Newton-Raphson method to solve a transcendental equation

Find a root of the equation 3x = cos x+1, near 1, by Newton Raphson method. Perform 5 iterations

```
from sympy import *
x= Symbol ('x')
g = input ('Enter the function ') #%3x -cos(x)-1; % function
f= lambdify (x,g)
dg = diff (g);
df= lambdify (x,dg)
x0= float ( input ('Enter the intial approximation ')); # x0=1
n= int( input ('Enter the number of iterations ')); #n=5;
for i in range (1,n+1):
    x1 =(x0 - (f(x0)/df(x0)))
    print ('itration %d \t the root %0.3f \t function value %0.3f \n'%(i, x1 ,f(x1))); # print all iteration value
x0 = x1
```

**6.4 Exercise:**

1. Find a root of the equation 3x = cos x+1, between 0 and 1, by Regula-falsi method. Perform 5 iterations.

Ans: 0.607

2. Find a root of the equation $xe^x = 2$, between 0 and 1, by Regula-falsi method. Correct to 3 decimal places.

Ans: 0.853

3. Obtain a real positive root of $x^4 - x = 0$, near 1, by Newton-Raphson method. Perform 4 iterations.
Ans: 1.856

4. Obtain a real positive root of $x^4 + x^3 - 7x^2 - x + 5 = 0$, near 3, by Newton-Raphson method. Perform 7 iterations.
Ans: 2.061

## LAB 7: Interpolation /Extrapolation using Newton's forward and backward difference formula

7.1 Objectives:

Use python
1. to interpolate using Newton's Forward interpolation method.
2. to interpolate using Newton's backward interpolation method.
3. to extrapolate using Newton's backward interpolation method.

1.Given f(40)=184, f(50)=204, f(60)=226, f(70)=250, f(80)=276, f(90)=304, find f(38) and f(85) using Newton's forward interpolation method.

```
import numpy as np
# Given data points
x = np.array([40, 50, 60, 70, 80, 90])
y = np.array([184, 204, 226, 250, 276,304])
coefficients = np.polyfit(x, y, deg=len(x) - 1)
def interpolate(x_val):
 return np.polyval(coefficients, x_val)
x_val_1 = 38
x_val_2 = 85
f_val_1 = interpolate(x_val_1)
f_val_2 = interpolate(x_val_2)
print("f(38) =", round(f_val_1,4)
print("f(85) =", round(f_val_2,4)
```

**ANS:**

```
f(38) = 180.24
f(85) = 289.75
```

2. Use Newtons interpolation to obtain the interpolating polynomial andhence calculate y(8) for the following data:

| | | | | | |
|---|---|---|---|---|---|
| x: | 1 | 3 | 5 | 7 | 9 |
| y: | 6 | 10 | 62 | 210 | 502 |

```
import numpy as np

# Given data points

x = np.array([1, 3, 5, 7, 9])

y = np.array([6, 10, 62, 210,502])

coefficients = np.polyfit(x, y, deg=len(x) - 1)

def interpolate(x_val):

 return np.polyval(coefficients, x_val)

x_val_1 = 8

f_val_1 = interpolate(x_val_1)

print("f(8) =", round(f_val_1,4))
```

**ANS :** `f(8) = 335.0`

**7.2 Exercise:**

1. Obtain the interpolating polynomial for the following data
x: 0 1 2 3
y: 1 2 1 10

Ans: $2x^3 - 7x^2 + 6x + 1$

2. Find the number of men getting wage Rs. 100 from the following table:
wage: 50 150 250 350
No. of men: 9 30 35 42

Ans: 23 men

3. Using Newton's backward interpolation method obtain y(160) for the following data
x : 100 150 200 250 300
y : 10 13 15 17 18

Ans: 13.42

4. Using Newtons forward interpolation polynomial and calculate y(1) and y(10).
x : 3 4 5 6 7 8 9
y : 4.8 8.4 14.5 23.6 36.2 52.8 73.9

Ans: 3.1 and 100

## LAB 8: Computation of area under the curve using Trapezoidal, Simpson's 1/3rd and simpson's 3/8th rule

8.1 Objectives:

Use python
1. to find area under the curve represented by a given function using Trapezoidal rule.
2. to find area under the curve represented by a given function using Simpson's 1/3rd rule.
3. to find area under the curve represented by a given function using Simpson's 3/8th rule.
4. to find the area below the curve when discrete points on the curve are given.

8.2 Trapezoidal Rule

Evaluate $\int_0^5 \frac{1}{1+x^2}$ dx

```
# Trapezoidal Rule
from sympy import *
x=Symbol('x')
g=input('Enter the function \t')
f=lambdify(x,g)
a=float(input('Enter lower limit \t'))
b=float(input('Enter upper limit \t'))
n=int(input('Enter the number of sub intervals'))
h=(b-a)/n
I=f(a)+f(b)
for j in range (1,n):
  i=a+j*h
  I=I+2*f(i)
I=(h/2)*I
print('Value of the integration corrected to four decimal places is', round(I, 4)))
```

**Output:**

```
Enter the function     1/(1+x**2)
Enter lower limit      0
Enter upper limit      1
Enter the number of sub intervals10
Value of the integration corrected to four decimal places is 0.785
```

### 8.3 Simpson's 1/3rd rule

Evaluate $\int_0^5 \frac{1}{1+x^2}$ dx with 100 subintervals.

```
# Simpson's (1/3)rd Rule
from sympy import *
x=Symbol('x')
g=input('Enter the function \t')
f=lambdify(x,g)
a=float(input('Enter lower limit \t'))
b=float(input('Enter upper limit \t'))

n=int(input('Enter the number of sub intervals \t'))
h=(b-a)/n

I=f(a)+f(b)

for j in range (1,n):
  i=a+j*h
  if j%2==0:
    I=I+2*f(i)
  else:
    I=I+4*f(i)
I=(h/3)*I

print('Value of the integration corrected to four decimal places is', round(I, 4))
```

**Output:**

```
Enter the function      1/(1+x**2)
Enter lower limit       0
Enter upper limit       1
Enter the number of sub intervals      10
Value of the integration corrected to four decimal places is 0.7854
```

**8.4 Simpson's 3/8th rule**

Evaluate $\int_0^5 \frac{1}{1+x^2}$ using Simpson's **3/8th** rule, taking 6 sub intervals.

```
# Simpson's (3/8)rd Rule
from sympy import *
x=Symbol('x')
g=input('Enter the function \t')
f=lambdify(x,g)
a=float(input('Enter lower limit \t'))
b=float(input('Enter upper limit \t'))

n=int(input('Enter the number of sub intervals \t'))
h=(b-a)/n

I=f(a)+f(b)

for j in range (1,n):
  i=a+j*h
  if j%3==0:
    I=I+2*f(i)
  else:
    I=I+3*f(i)
I=(3*h/8)*I

print('Value of the integration corrected to four decimal places is, round(I, 4))
```

**Output:**

```
Enter the function     1/(1+x**2)
Enter lower limit      0
Enter upper limit      1
Enter the number of sub intervals     6
Value of the integration corrected to four decimal places is 0.7854
```

8.5 Exercise:

1. Evaluate the integral $\int_0^1 \frac{x^2}{1+x^3} dx$ using Simpson's 1/3rd Rule

Ans: 0.23108

2. Use Simpson's 3/8th $\int_0^{0.6} e^{-x^2} dx$ rule to find by taking seven ordinates.

Ans: 0.5351

3. Evaluate using trapezoidal rule $\int_0^{\pi} sin^2 x dx$Take n = 6.

Ans: π/2

4. A solid of revolution is formed by rotating about the x-axis, the area between the x-axis, the lines x = 0 and x = 1, and a curve through the points with the following co-ordinates:

| x | y |
|---|---|
| 0.00 | 1.0000 |
| 0.25 | 0.9896 |
| 0.50 | 0.9589 |
| 0.75 | 0.9089 |
| 1.00 | 0.8415 |

Estimate the volume of the solid formed using Simpson's 1/3rd rule.

Hint: Required volume is $\int_0^1 y^2 * \pi dx$ **[Ans: 2.8192]**

5. The velocity v(km/min) of a moped which starts from rest, is given at fixed intervals of time t(min) as follows:

t: 2 4 6 8 10 12 14 16 18 20

v: 10 18 25 29 32 20 11 5 2 0

Estimate approximately the distance covered in twenty minutes.

## LAB 9: Solution of ODE of first order and first degree by Taylor's series and Modified Euler's method

9.1 Objectives:

Use python

1. to solve ODE by Taylor series method.
2. to solve ODE by Modified Euler method.

9.2 Taylor series method to solve ODE

1.From Taylor series method find y(0.1) for $\frac{dy}{dx} = x^2 + y^2$ at x=0.1,0.2 given $x0 = 0, y0 = 1$ upto third degree.

```
from sympy import *
x,y=symbols('x,y')
x0,x1,x2=0,.1,.2
yd0=1
yd1=x**2+y**2
yd2=2*x+2*y*yd1
yd3=2+2*y*yd2+yd1*2*yd1
yd1=yd1.subs({x:0,y:1})
yd2=yd2.subs({x:0,y:1})
yd3=yd3.subs({x:0,y:1})
ts=yd0+(x-x0)*yd1+(x-x0)**2*yd2/2+(x-x0)**3*yd3/6
tsx1=ts.subs(x,x1)
tsx2=ts.subs(x,x2)
display('Taylors series is',ts)
print('y(',x1,')=',round(tsx1,4))
print('y(',x2,')=',round(tsx2,4))
```

ANS:

```
Taylors series is 4*x**3/3 + x**2 + x + 1
y( 0.1 )= 1.1113
y( 0.2 )= 1.2507
```

b) Taylor's series method to solve dy/dx=x+y at x= 1.1, 1.2 given x0=1,y0=0 up to the third degree terms

```
from sympy import *
x,y=symbols('x,y')
x0,x1,x2=1,1.1,1.2
yd0=0
yd1=x+y
yd2=1+yd1
yd3=yd2
yd1=yd1.subs({x:1,y:0})
yd2=yd2.subs({x:1,y:0})
yd3=yd3.subs({x:1,y:0})
ts=yd0+(x-x0)*yd1+(x-x0)**2*yd2/2+(x-x0)**3*yd3/6
tsx1=ts.subs(x,x1)
tsx2=ts.subs(x,x2)
display('Taylors series is',ts)
print('y(',x1,')=',round(tsx1,4))
print('y(',x2,')=',round(tsx2,4))
```

ANS :

```
Taylors series is x + (x - 1)**3/3 + (x - 1)**2 - 1
y( 1.1 )= 0.1103
y( 1.2 )= 0.2427
```

## 9.3 Modified Euler's method

The iterative formula is:

$$y_1^{(n+1)} = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(n)})], \qquad n = 0, 1, 2, 3, \ldots,$$

where, $y_1^{(n)}$ is the $n^{th}$ approximation to $y_1$.

1 . Euler's modified method to solve dy/dx=x-y**2 at x=0.0(0.1)0.2 given x0=0,y0=1

```
def euler(x0, x1, y0, h):
    def f(x, y):
        return x - y**2
    n = round((x1 - x0) / h)
    for i in range(1, n + 1):
        y1 = y0 + h * f(x0, y0)
        print('y(', round(x0 + h, 3), ') =', round(y1, 4))
        for j in range(1, 4):
            y1 = y0 + h / 2 * (f(x0, y0) + f(x0 + h, y1))
            print('The', j, 'th improved value of y(', round(x0 + h, 3), ') =', round(y1, 4))
        x0, y0 = x0 + h, y1
euler(0.0, 1, 1, 0.2)
```

**ANS :**

```
y( 0.2 ) = 0.8
The 1 th improved value of y( 0.2 ) = 0.856
The 2 th improved value of y( 0.2 ) = 0.8467
The 3 th improved value of y( 0.2 ) = 0.8483
y( 0.4 ) = 0.7444
The 1 th improved value of y( 0.4 ) = 0.7809
The 2 th improved value of y( 0.4 ) = 0.7754
The 3 th improved value of y( 0.4 ) = 0.7762
y( 0.6 ) = 0.7357
The 1 th improved value of y( 0.6 ) = 0.7618
The 2 th improved value of y( 0.6 ) = 0.7579
The 3 th improved value of y( 0.6 ) = 0.7585
y( 0.8 ) = 0.7635
The 1 th improved value of y( 0.8 ) = 0.7827
The 2 th improved value of y( 0.8 ) = 0.7797
The 3 th improved value of y( 0.8 ) = 0.7802
y( 1.0 ) = 0.8185
The 1 th improved value of y( 1.0 ) = 0.8323
The 2 th improved value of y( 1.0 ) = 0.83
The 3 th improved value of y( 1.0 ) = 0.8304
```

9.4 Exercise:

1. Find y(0.1) by Taylor Series exapnsion when $y' = x - y^2$, y(0) = 1.
Ans: y(0.1) = 0.9138

2. Find y(0.2) by Taylor Series exapnsion when $y' = x^2y - 1$, y(0) = 1, h = 0.1.
Ans: y(0.2) = 0.80227

3. Evaluate by modified Euler's method: $y' = \ln(x + y)$, y(0) = 2 at x = 0(0.2)0.8.
Ans: 2.0656, 2.1416, 2.2272, 2.3217

4. Solve by modified Euler's method: $y' = x + y$, y(0) = 1, h = 0.1, x = 0(0.1)0.3.
Ans: 1.1105, 1.2432, 1.4004

## LAB 10: Solution of ODE of first order and first degree by Runge-Kutta 4th order method and Milne's predictor and corrector method

10.1 Objectives:

1. To write a python program to solve first order differential equation using 4th order Runge Kutta method.
2. To write a python program to solve first order differential equation using Milne's predictor and corrector method.

10.2 Runge-Kutta method

1.Apply the Runge Kutta method to find the solution of dy/dx = 1 + (y/x) at 0.4 by taking h = 0.2. Given that y(1) = 2.

```
def rk(x0, x1, y0, h):
    def f(x, y):
        return x + y**2
    n = round((x1 - x0) / h)
    for i in range(1, n + 1):
        k1 = h * f(x0, y0)
        k2 = h * f(x0 + h / 2, y0 + k1 / 2)
        k3 = h * f(x0 + h / 2, y0 + k2 / 2)
        k4 = h * f(x0 + h, y0 + k3)
        k = (k1 + 2 * k2 + 2 * k3 + k4) / 6
        print('k1 =', round(k1, 4), 'k2 =', round(k2, 4), 'k3 =', round(k3, 4), 'k4 =', round(k4, 4), 'k =', round(k,4))
        y1 = y0 + k
        print('y(', round(x0 + h, 3), ') =', round(y1, 4))
        x0, y0 = x0 + h, y1
rk(0, 0.4, 1, 0.2)
```

**ANS :**

```
k1 = 0.2 k2 = 0.262 k3 = 0.2758 k4 = 0.3655 k = 0.2735
y( 0.2 ) = 1.2735
k1 = 0.3644 k2 = 0.4838 k3 = 0.5193 k4 = 0.7229 k = 0.5156
y( 0.4 ) = 1.7891
```

**2.** Runge - Kutta fourth order method to solve dy/dx=3*x + y/2at x=0.2(0.2)0.4 given x0=0,y0=1

```
def rk(x0, x1, y0, h):
    def f(x, y):
        return 3*x + y/2
    n = round((x1 - x0) / h)
    for i in range(1, n + 1):
        k1 = h * f(x0, y0)
        k2 = h * f(x0 + h / 2, y0 + k1 / 2)
        k3 = h * f(x0 + h / 2, y0 + k2 / 2)
        k4 = h * f(x0 + h, y0 + k3)
        k = (k1 + 2 * k2 + 2 * k3 + k4) / 6
        print('k1 =', round(k1, 4), 'k2 =', round(k2, 4), 'k3 =', round(k3, 4), 'k4 =', round(k4, 4), 'k =',
round(k,4))
        y1 = y0 + k
        print('y(', round(x0 + h, 3), ') =', round(y1, 4))
        x0, y0 = x0 + h, y1
rk(0, 0.4, 1, 0.2)
```

**ANS:**

```
k1 = 0.1 k2 = 0.165 k3 = 0.1683 k4 = 0.2368 k = 0.1672
y( 0.2 ) = 1.1672
k1 = 0.2367 k2 = 0.3086 k3 = 0.3121 k4 = 0.3879 k = 0.311
y( 0.4 ) = 1.4782
```

## Milne's predictor and corrector method

Apply Milne's predictor and corrector method to solve $\frac{dy}{dx} = x - y^2$ . Given that y(0)=0, y(0.2)=0.02, y(0.4)=0.0795, y(0.6)=0.1762. compute y(0.8).

```
def f(x, y):
    F = x - y ** 2
    return F
x0 = 0
y0 = 0
h = 0.2
# Assuming y1, y2, and y3 (required for Milne formula) are estimated
x1 = x0 + h
y1 = 0.02
x2 = x1 + h
y2 = 0.0795
x3 = x2 + h
y3 = 0.1762
# Milne Predictor formula
yp4 = y0 + 4 * h * (2 * f(x1, y1) - f(x2, y2) + 2 * f(x3, y3)) / 3
x4 = x3 + h
fp4 = f(x4, yp4)
print('yp4 =', yp4)
print('fp4 =', fp4)
# Corrector formula
yc4 = y2 + h * (f(x2, y2) + 4 * f(x3, y3) + fp4) / 3
f4 = f(x4, yc4)
print('yc4 =', yc4)
print('f4 =', f4)
yc4 = y2 + h * (f(x2, y2) + 4 * f(x3, y3) + f4) / 3
print('yc4 =', yc4)
```

**ANS:**

```
yp4 = 0.3049139653333334
fp4 = 0.7070274737447028
yc4 = 0.30460143091631353
f4 = 0.7072179682837343
yc4 = 0.30461413055224895
```

**10.4 Exercise:**

1. Find y(0.1) by Runge Kutta method when $y' = x - y^2$, y(0) = 1.
Ans: y(0.1) = 0.91379

2. Evaluate by Runge Kutta method : $y' = \log(x + y)$, y(0) = 2 at x = 0(0.2)0.8.
Ans: 2.155, 2.3418, 2.557, 2.801

3. Solve by Milnes method: $y' = x + y$, y(0)=1, h=0.1, Calculate y(0.4) . Calculate required initial values from Runge Kutta method.
Ans: 1.583649219