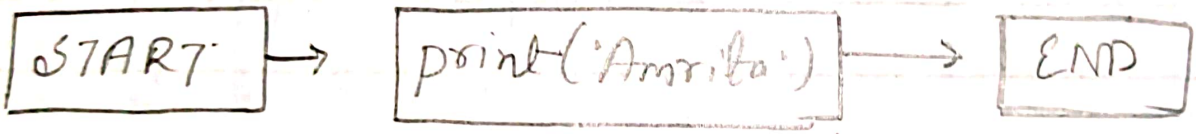
Experiment NO.→1

Aim : Implementation of python Basics : variable, Print Statement, Commands, Data Types : Numeric Types, String, List and Tuple, Control Structures : If, For and while, Functions : Function Definition, Function Call, Return Statement, Default parameters.

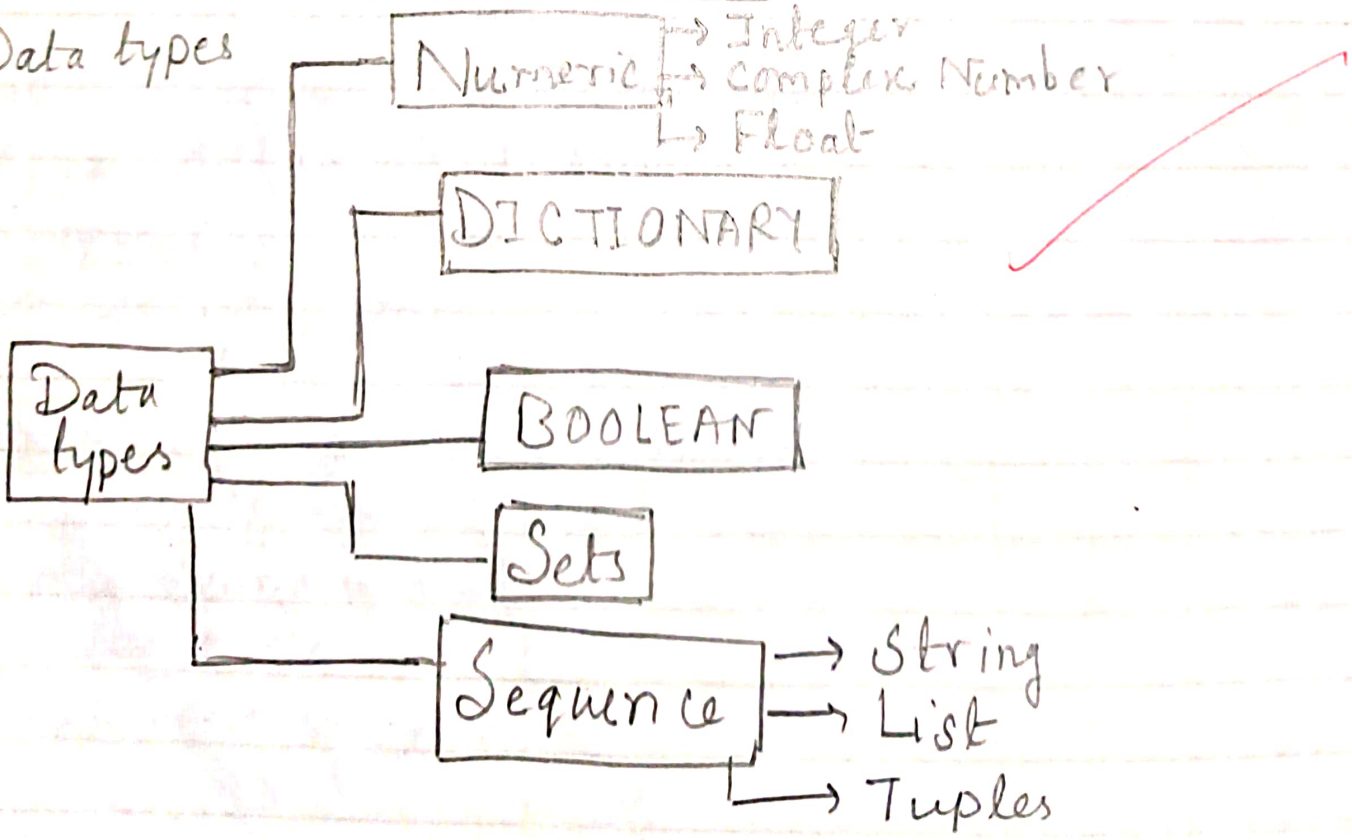
Software Requirement : Google colab

Flowchart :

① Print Statement



② Data types



Date _____ Expt. No. 01

Expt. Name _____ Page No. 02

Aim : Implementation of Python Basics : Variable, print Statements, commands, Data types : Numeric Types, String, List and tuple, control Structures : If, for and while, functions : function Definition, Function call, Return statement, Default parameters.

Software Requirement : Google Colab

Description :

① Variable SYNTAX : variable_name = value

Variables are used to store data values in python. They are dynamically typed, meaning we can change the type of a variable by assigning a new value.

② Print Statement

SYNTAX : Print (objects, sep = ' ', end = '\n', flush = False)

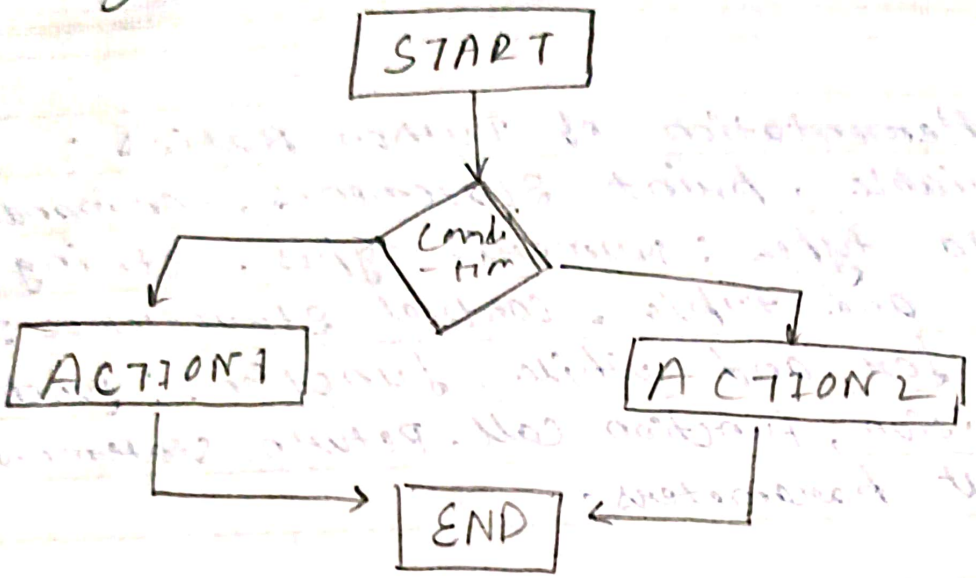
The 'Print' function is used to output text or other data to the console.

③ Datatypes :

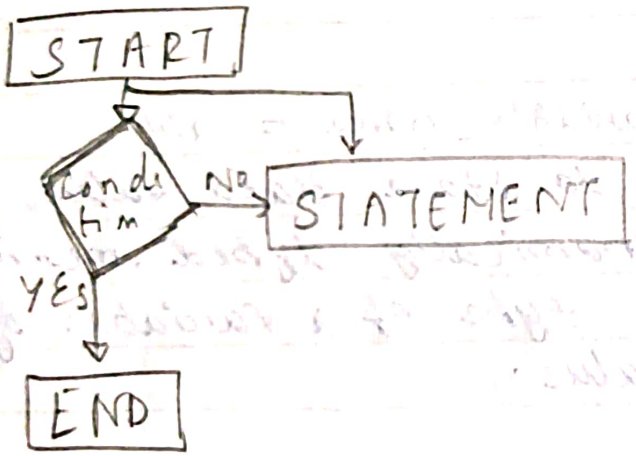
(i) Numeric

Teacher's Signature: _____

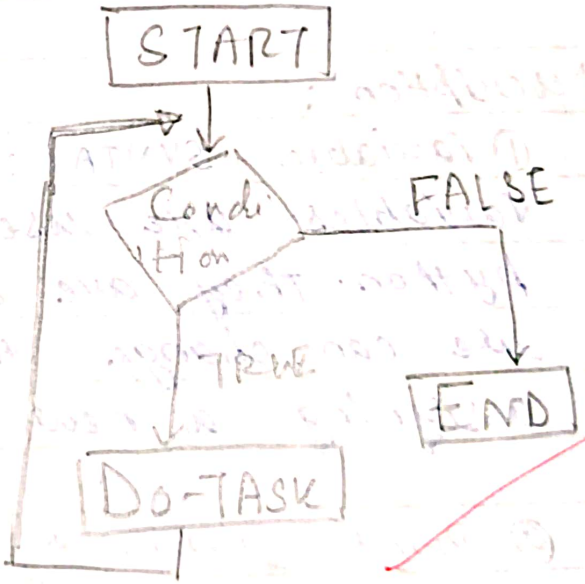
③ If Statement



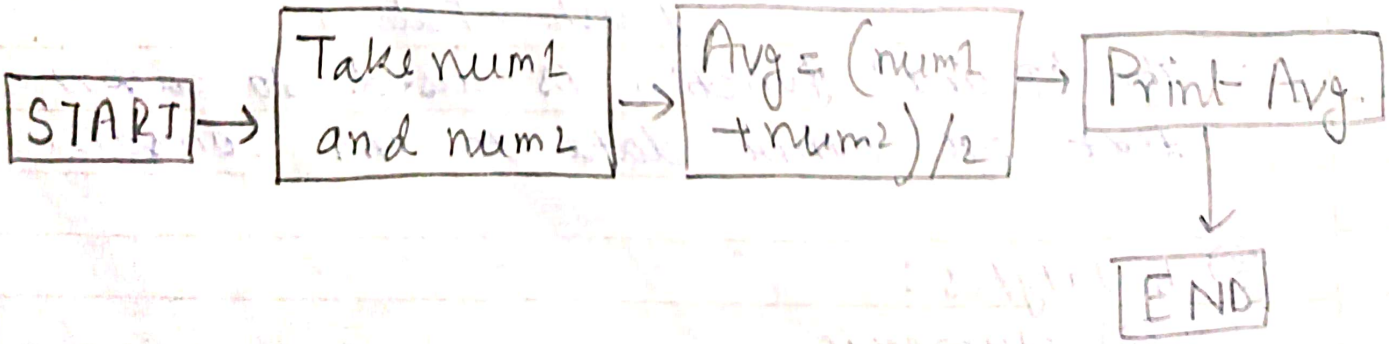
④ FOR LOOP



⑤ WHILE LOOP



⑥ Function Call



Date _____ Expt. No. _____

Expt. Name _____ Page No. 03

- Integer ('int') SYNTAX: x = 10
- Float ('float') SYNTAX: x = 10.5
- complex ('complex') SYNTAX: x = 3+5j

(ii) String

SYNTAX: x = "Amrita Ranjan"

(iii) List SYNTAX: x = [1, 2, 3]

(iv) Tuple SYNTAX: x = (1, 2, 3)

④ control Structure

(i) If Statement

The 'if' statement allows to execute a block of code based on a condition.

```
SYNTAX : if condition :
             # code_block
         elif another_condition :
             # code_block
         else :
             # code_block
```

(ii) For Loop

The 'for' loop is used to iterate over a sequence.

```
SYNTAX : for item in sequence :
             # code_block
```

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 04

(iii) While Loop

The 'while' loop allows to execute a block of code as long as a condition is true.

SYNTAX: while condition :
    # code_block

(iv) Break and continue Statements

We can use 'break' and 'continue' within loops to alter the flow of control.
'break' is used to exit a loop statement prematurely when a certain condition is met & 'continue' skips the rest of the code inside the loop for the current iteration and jumps to next iteration.

(v) Functions :

→ Function Definition

SYNTAX: def function_name (parameters):
    # code - block

→ Function Call

To call a function, use the function name followed by parentheses containing any arguments. SYNTAX: function_name (arguments)

→ Return Statement

This is used to exit a function & return a value to the caller. If no 'return' statement is present, the function returns

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 05

'None' by default.

SYNTAX : def function_name (parameters):
code_block
return block

→ Lambda function

This is defined using the 'lambda' keyword. Lamda keywords are often used for short, simple operation where defining a full function might be overkill.

SYNTAX : lambda arguments : expression.

Conclusion

Conclusion :

Through this experiment, we have learnt hands-on practice & real-world examples, the implementation of python basics will re-inforce your understanding and prepare you to move on to more advanced topics. This will help to build projects in python.

Teacher's Signature: _____

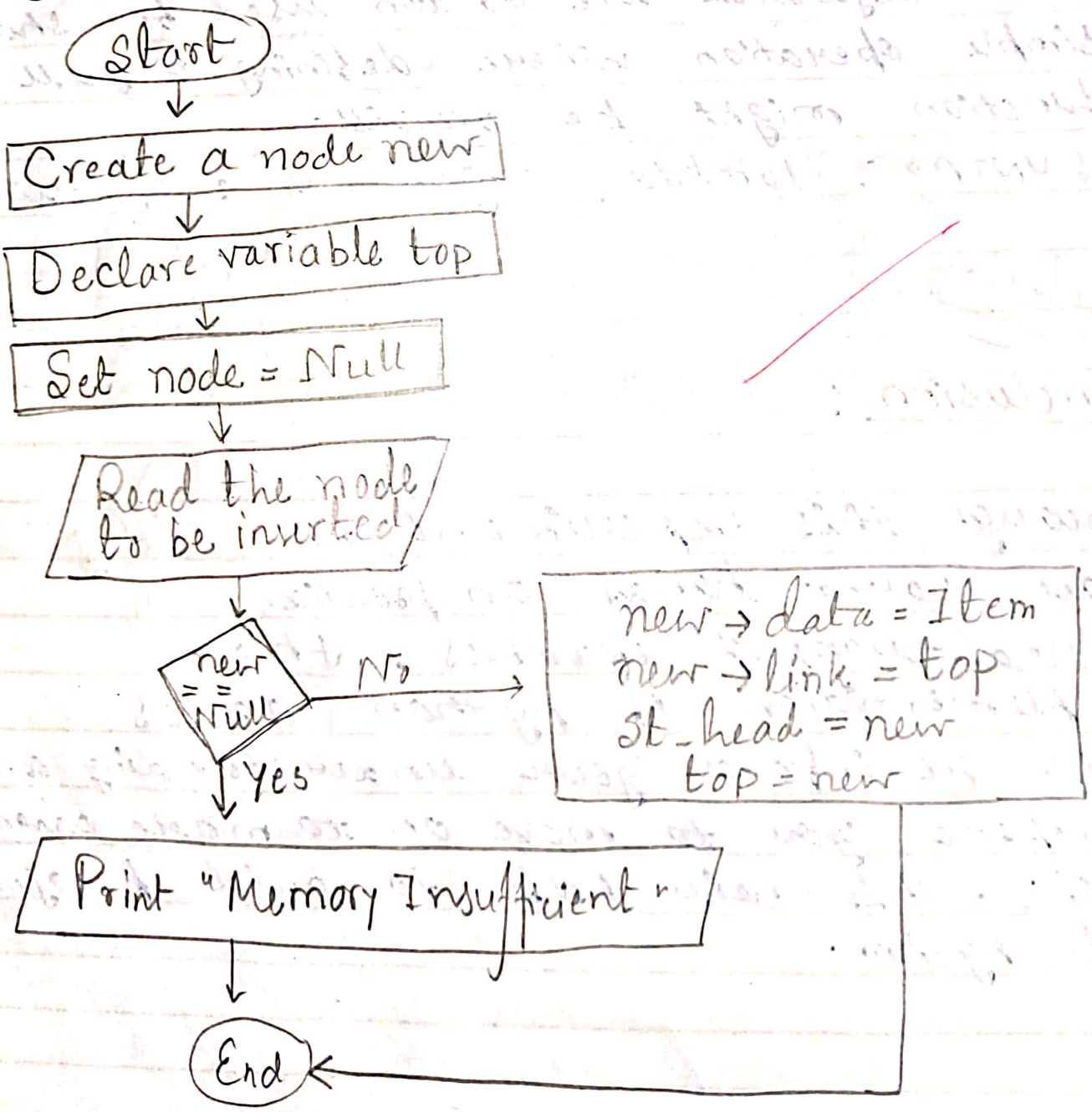
Experiment 1.2

Aim : Implementation of different data structures in python such as stack, queue and linked list.

Software Requirement : Google colab

Flowchart :

① Push()



Date 9/4/24 Expt. No. 02

Expt. Name _____ Page No. 06

Aim: Implementation of different data structures in python such as stack, queue and linked list.

Software Requirement: Google colab

Description:

Some descriptions about Stack, queue and linked list.

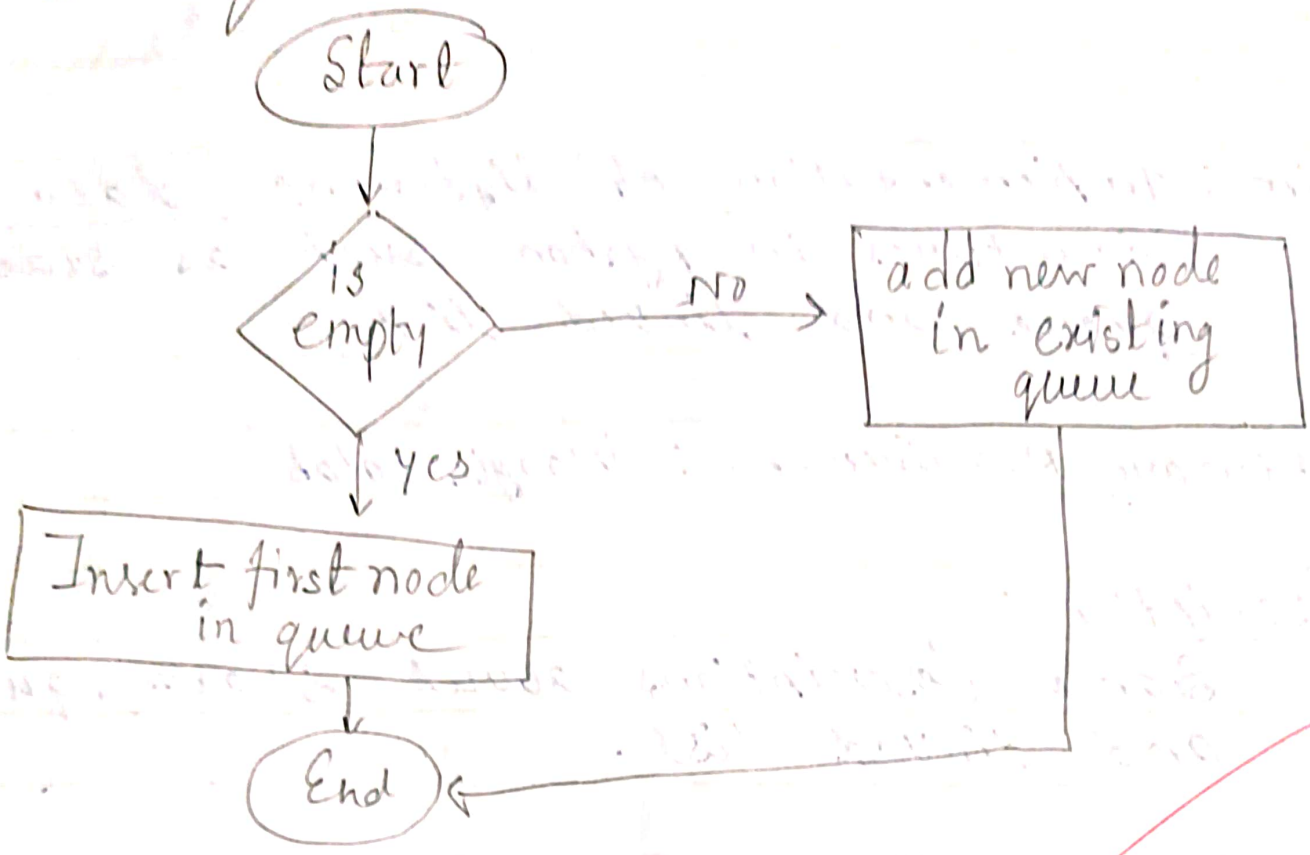
→ Stack :- It is a linear data structure that follows the last-In-First-out (LIFO) principle. This means that the last element added to the stack will be first one to be removed.

* Elements are added and removed from the top.
* Do random access.

→ Queue :- It is a linear data Structure that follows First-in-first-out (FIFO) principle. this means that line first element added to the queue will be the first element to be removed.

Teacher's Signature: _____

② enqueue()



Code and Output:

Date _____ Expt. No. _____

Expt. Name _____ Page No. 07

* Elements are added to the end and removed from the front.
* No random access.

→ Linked list : A linked list is a dynamic data structure consisting of nodes that are linked together by pointers. Each node contains a value and a reference to the next node in the list.

* Elements are stored in separate objects.
* Elements are linked together.

Algorithm :

→ Stack Implementation :-

i) initialise an empty list to represent stack.
ii) Define method i.e., Push, Pop & peek.
iii) Implement push to add element.
iv) Implement pop to delete last element and return top element.
v) Implement peek to return top element without removing.
vi) Implement method to check if stack is

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 08

empty.

→ Queue Implementation :-

i) Initialize empty list to represent queue.

ii) Define enqueue, dequeue and front.

iii) Implement enqueue to add element.

iv) Implement dequeue to remove and return last element.

v) Implement front to print front element.

vi) Implement a method to check if queue is empty.

→ Linked list implementation :-

i) Define node class to represent element in the linked list i.e., containing data and reference to next node.

ii) Define linked list to manage nodes.

iii) Implement insert, delete and display.

iv) Implement delete to remove elements.

Conclusion :

Through this experiment we successfully implement & understood the work of three fundamental data structure : Stack, queue & linked list. It is crucial for developing efficient algorithm.

Teacher's Signature: Saroj 6/8/24

Experiment → 1.3

Aim : Implementing Root - finding algorithms (Bisection method, Newton - Raphson method) in python.

Software Requirement : Google Colab

Flowchart :

① **Bisection method :**

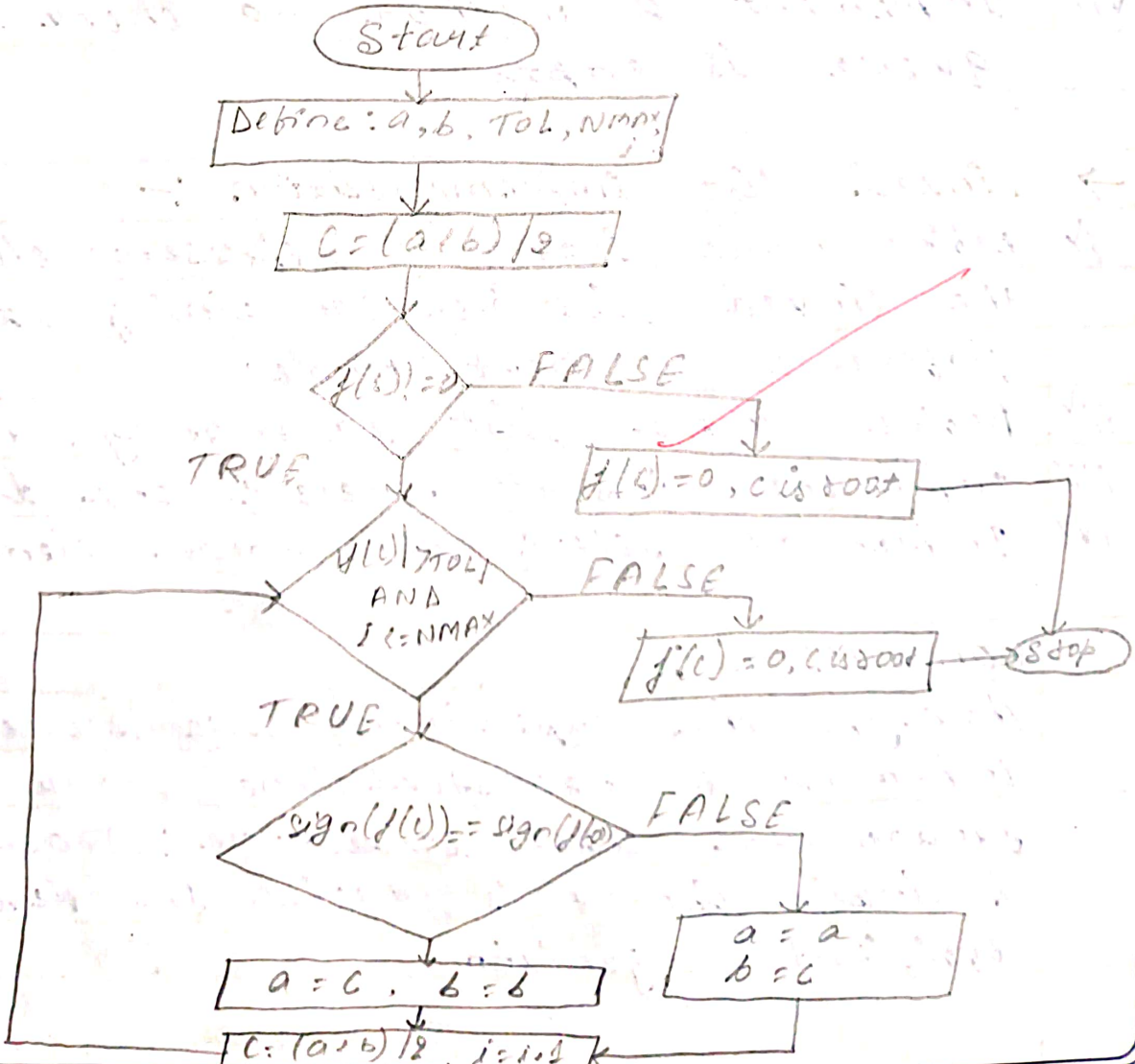


fig. 1

Date 16/8/24 Expt. No. 03

Expt. Name _____ Page No. 09

Aim: Implementing Root-finding Algorithms (Bisection method, Newton-Raphson method in Python.

Software Requirement → Google colab

Description:

Root finding algorithms are essential numerical methods used to determine where a given function $f(x)$ equals zero, commonly referred to as finding the roots of the equation $f(x) = 0$. Two widely use root-finding algorithms are the Bisection method and the Newton-Raphson method each with distinct approaches and characteristics.

The Bisection method is a straightforward & robust technique that relies on the Intermediate Value Theorem. It works by repeatedly narrowing an interval $[a, b]$ where the function changes sign, meaning $f(a) \times f(b) < 0$. By bisecting the interval and selecting the subinterval where the sign change occurs, the method systematically converges

Teacher's Signature: _____

② Newton - Raphson method :

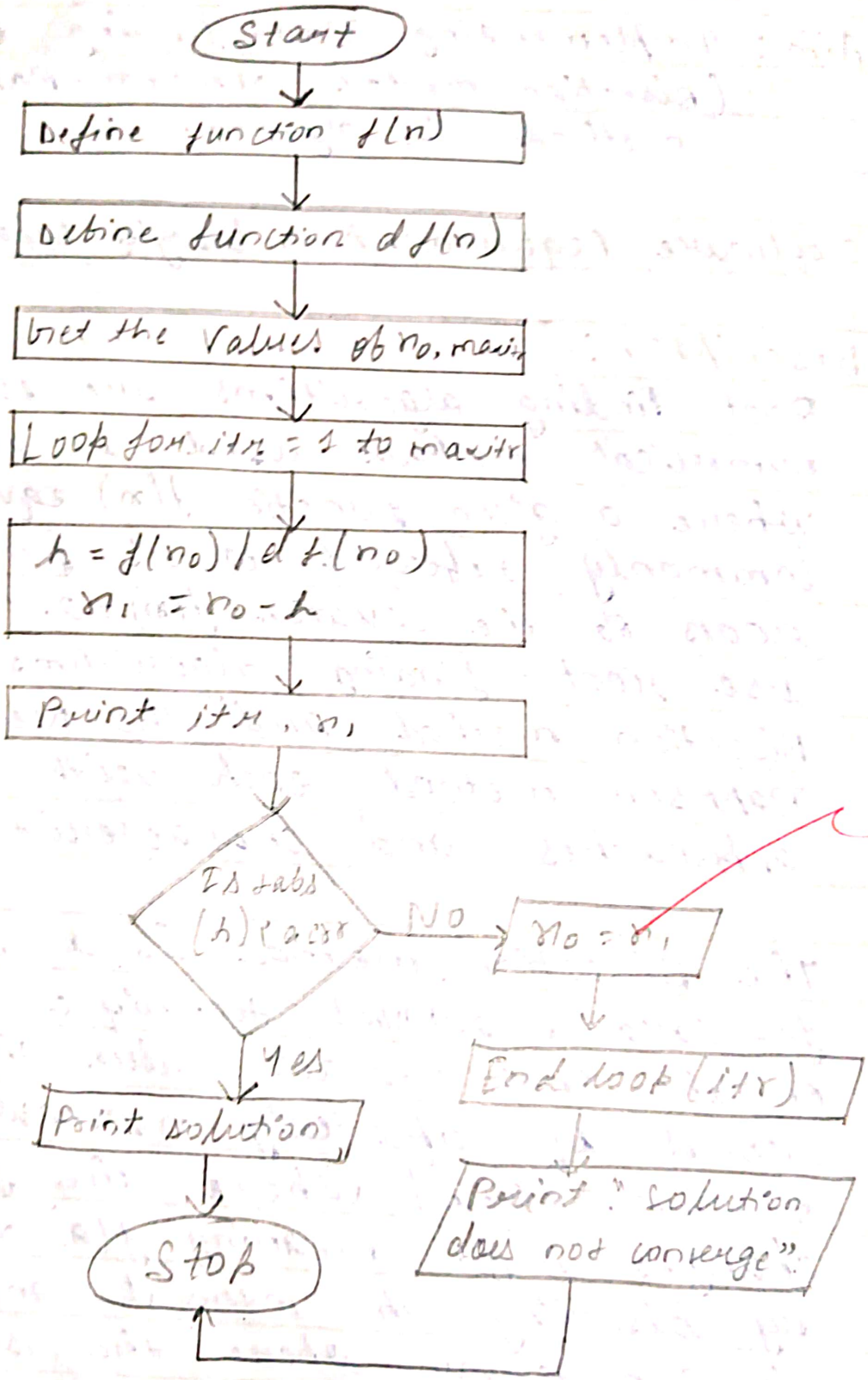


fig. 2

to the root. The Bisection method guarantees convergence, but it can be relatively slow compared to another methods, as it only have the interval size in each iteration.

On the other hand, the Newton-Raphson method is an iterative approach that typically converges much faster, especially when the initial guess is close to the actual root. This method uses the function's derivative to refine guesses through the formula,

$$x_{n+1} = x_n - f(x_n) / f'(x_n)$$

The Newton-Raphson method is more efficient, requiring fewer iterations to reach the desired accuracy. However, it has a downside: it depends on the function being differentiable and may fail to converge if the derivative is zero or undefined at any step.

Algorithm:

① Bisection method:

Step 1: Start

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 11

Step 2: Take two points, a and b, such that $f(a)$ and $f(b)$ have opposite signs, meaning $f(a) \times f(b) < 0$.

Step 3: Calculate / compute the midpoint
$c = (a+b)/2$.

Step 4: Evaluate $f(c)$.
If $f(c) = 0$, c is the root.
If not, determine the subinterval $[a, c]$ or $[c, b]$ where the sign changes & repeat the process.

Step 5: continue the process until the interval $[a, b]$ is sufficiently small i.e., the difference $b-a$ is less than a given tolerance.

Step 6: End.

(2) Newton - Raphson method:

Step 1: Start

Step 2: Start with an initial guess x_0.

Step 3: Improve the guess using formula:
$x_{n+1} = x_n - f(x_n) / f'(x_n)$

Step 4: Repeat the iteration until the difference between consecutive guesses is smaller than a given tolerance.

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 12

Step 5 : End.

Learning outcome:

1. Understand the numerical methods.
2. Learn algorithmic problem solving.
3. Learn how to apply concepts from Calculus.
4. Learn about different Numerical Optimisation methods.
5. Learn how to find roots in Bisection method and Newton-Raphson method.

Experiment: 4

Aim: Implementing Secant algorithm in Python.

Software Required: Google Colab.

Flowchart:

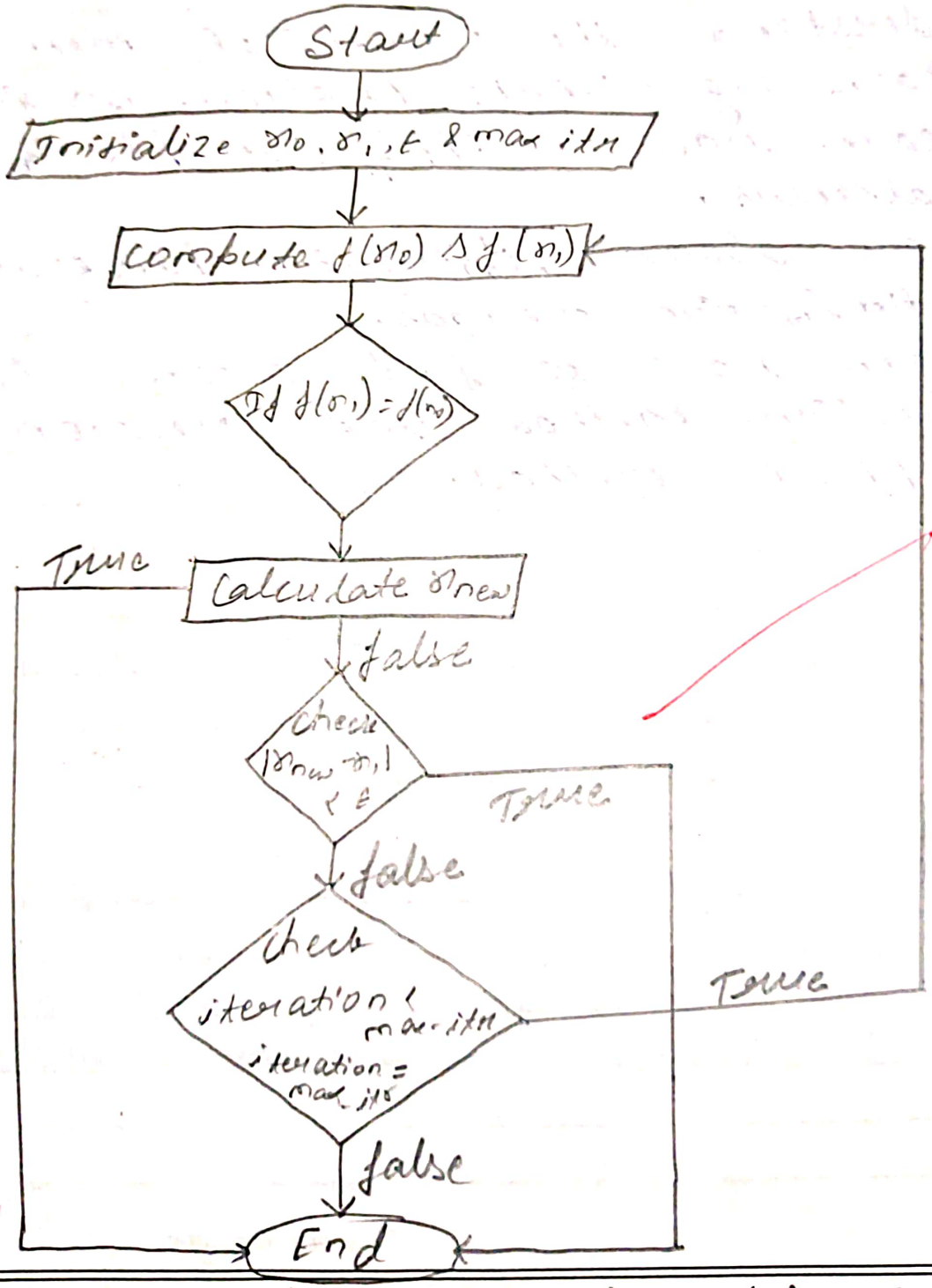


fig. 1: Flowchart of Secant Algo.

Date 28/8/24 Expt. No. 1.4

Expt. Name Page No. 13

Aim: Implementing Secant algorithm in Python.

Software Requirement: Google colab

Description:

The Secant algorithm is a root-finding algorithm that approximates the root of a function by using a sequence of Secant lines rather than relying on the function's derivative, as in the Newton-Raphson method. It uses two initial guesses, x_0 & x_1 and iteratively refines these guesses to approximate the root of the function $f(x) = 0$.

The formula used in this method is:-

$$x_{n+1} = x_n - f(x_n) \times \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Algorithm:

1) Start with two initial guesses x_0 & x_1.
2) Compute the function values at these points: $f(x_0)$ & $f(x_1)$

Teacher's Signature:

Date _____ Expt. No. _____

Expt. Name _____ Page No. 14

3) Use the formula: $x_2 = x_1 - f(x_1) \times \frac{x_1 - x_0}{f(x_1) - f(x_0)}$

4) Set x_0 to x_1 & x_1 to x_2 for the next iteration.

5) Check if the difference $|x_2 - x_1|$ is less than a given tolerance tol. If so, the algorithm has converged & x_2 is the root.

6) If the maximum number of iterations is reached without satisfying the tolerance, the algorithm may terminate and indicate that convergence was not achieved.

7) Repeat the process until root is found within the desired tolerance or the maximum number of iterations is reached.

8) Stop / End.

Learning outcome:

1) Understood Numerical methods from Secant algorithm.

2) Learn algorithmic problem solving.

3) Learn how to apply concepts from calculus.

Teacher's Signature: _____ 6/9/24.

Experiment - 2.1

Aim: Implement hands-on interpolation exercises using python libraries.

Software Requirement → Python 3.11

Flowchart →

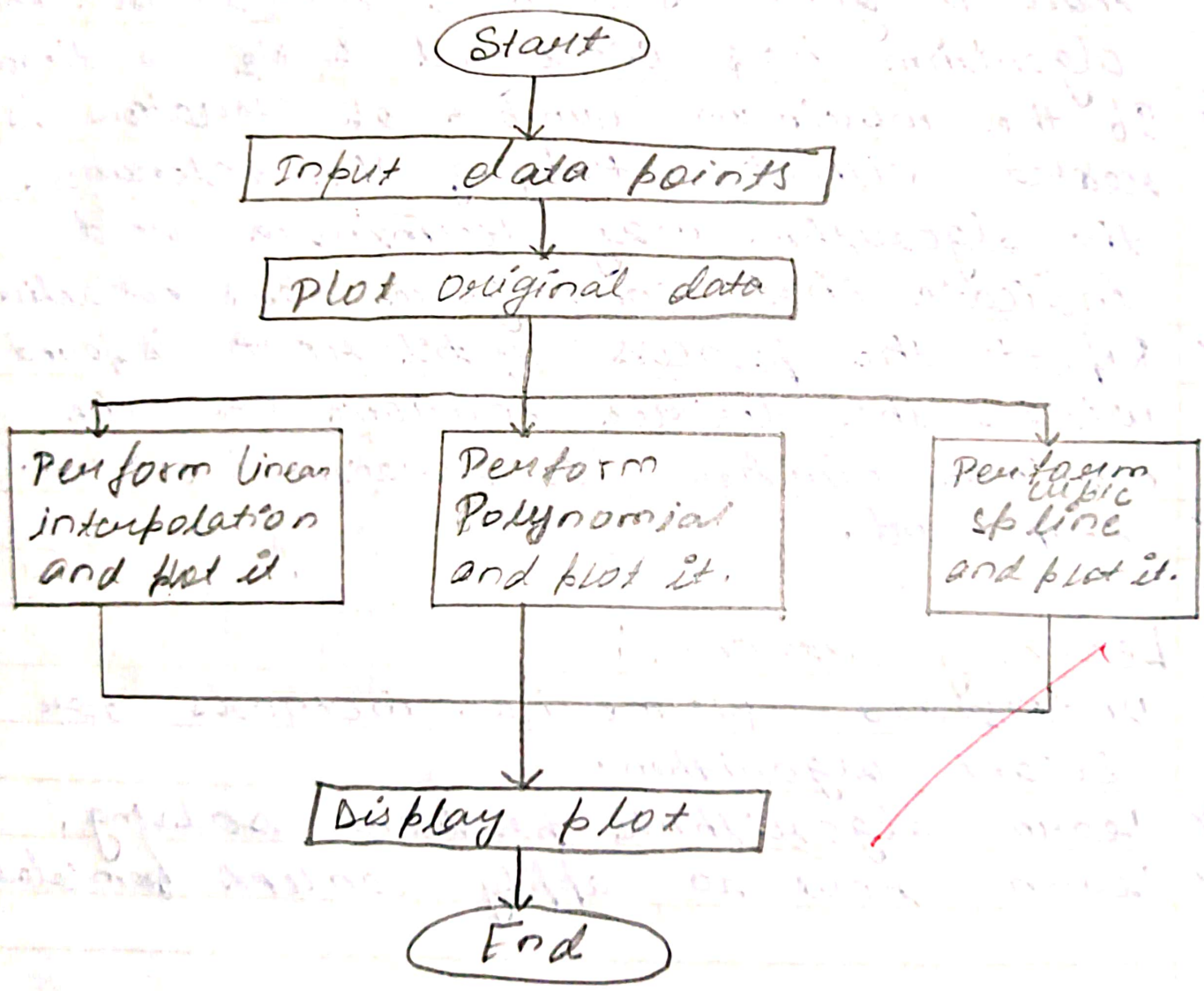


fig. 1 : flowchart

Date 13/9/24 Expt. No. 2.1

Expt. Name _____ Page No. 15

Aim → Hands on Interpolation exercises using python libraries.

Software Requirement → Python 3.11

Description → It is a method used to estimate unknown values between two known values. In mathematical, statistical and computational contexts. Interpolation is widely used to approximate function, data points or curves.

Types →

1.) Linear Interpolation:

The simplest form of interpolation, where you estimate value of a function by assuming that value changes linearly b/w two known points.

$$y = y_0 + \frac{(y_1 - y_0)(x - x_0)}{(x_1 - x_0)}$$

11.) Polynomial Interpolation:

It is fitted data points and interpolated values are estimated using polynomial equation.

$$P(x) = \sum_{i=0}^{n-1} y_i \prod_{\substack{0 \le j \le n-1 \\ i \neq 1}} \frac{x - x_j}{x_i - x_j}$$

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 16

iii) Spline Interpolation → It is a piece-wise defined polynomial, usually cubic, that is fitted b/w each pair of data points.

$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)$.

Algorithm →

Step 1) Start.

Step 2) Initialize libraries such as numpy, scipy.

Step 3) Input data

Step 4) Perform Interpolation :- (i) Linear Interpolation
(ii) Polynomial "
(iii) Spline Interpolation.

Step 5) Customize and plot all Interpolation.

Step 6) End.

Learning outcomes →

i) Understanding interpolation concepts.

ii) Practical application of python libraries.

iii) How to handle numerical data.

Teacher's Signature: Saroj 1/7/24

Experiment-2.2

Aim: Implementing two common numerical integration algorithms, the trapezoidal rule and Simpson's rule.

Software Requirement: Google Colab.

Flowchart:

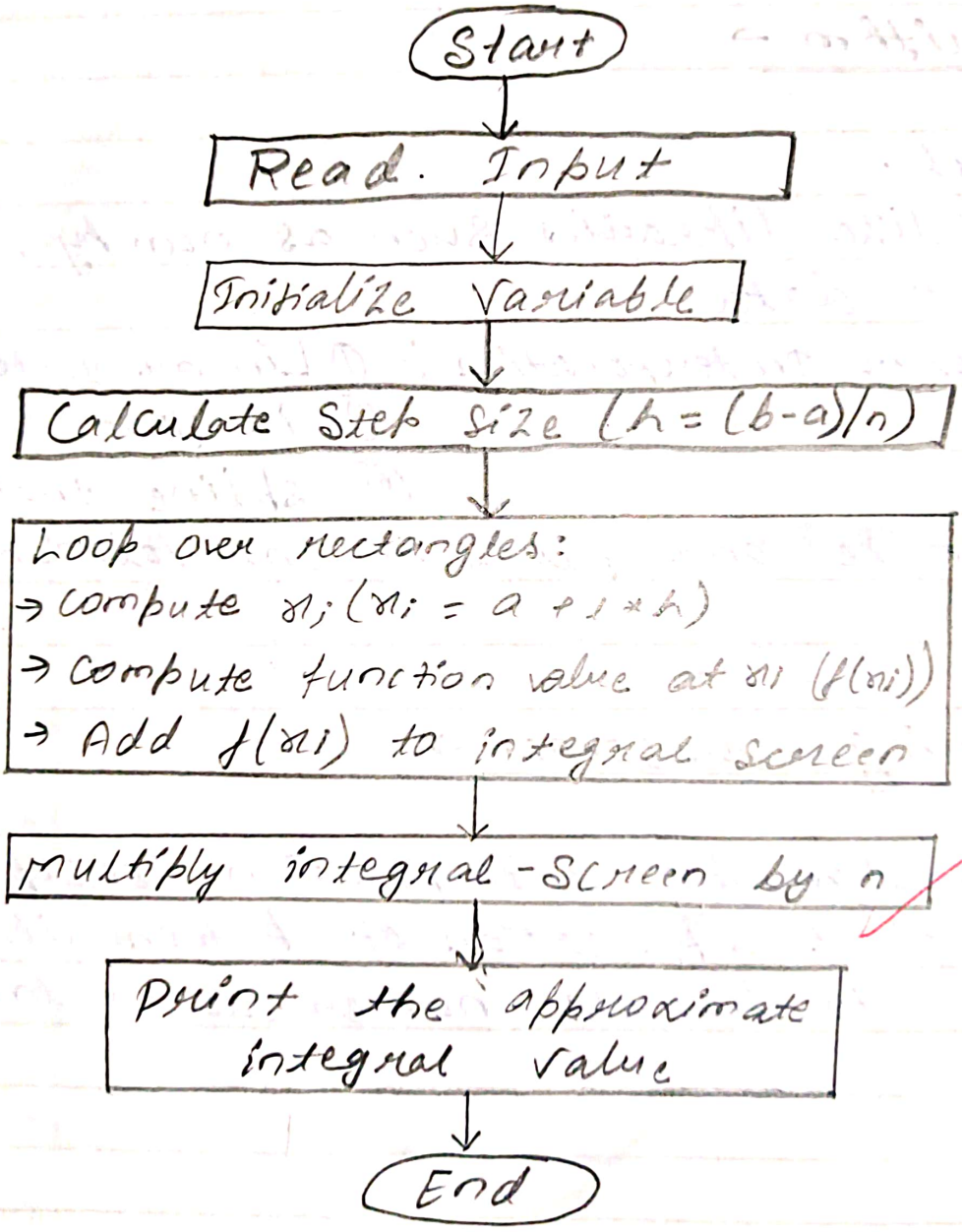


Fig.1: Flowchart of Rules

Date 27/9/24 Expt. No. 2.2

Expt. Name Page No. 17

Aim: Implementing two common numerical integration algorithms. the trapezoidal rule and Simpson's rule.

Software Requirement: Google Colab

Description: The trapezoidal rule is a numerical method for approximating the definite integral of a function. It works by dividing the area under the curve into a series of adjacent trapezoids, rather than rectangles, to better approximate the curve.
Given a function $f(x)$ over the interval $[a, b]$ divided into n subintervals:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)\right]$$

where,

$h = \frac{b-a}{n}$ is the width of each subinterval

Simpson's rule is a more accurate numerical integration method that approximates the function using parabolic (quadratic) segments instead of straight lines.

Teacher's Signature:

Date _____ Expt. No. _____

Expt. Name _____ Page No. 18

For a $\int^{n} f(x)$ over interval $[a, b]$:

$$\int_a^b f(x)dx \simeq h/3 \left[f(a) + 4 \sum_{i=1}^{n-1} f(x_i) + 2 \sum_{i=2}^{n-1} f(x_i) + f(b) \right]$$

Algorithm:

Trapezoidal Rule code:

Step 01: Calculate the width of each interval.

Step 02: Generate Discrete points.

Step 03: Calculate or evaluate function.

Step 04: Apply Rule and Return the Result.

Simpson's Rule code:

Step 01: check Interval count

Step 02: Calculate the width of each interval.

Step 03: Generate Discrete points.

Step 04: Evaluate the function.

Step 05: Apply Rule and Return result.

Learning Outcome:

i) Implement the trapezoidal rule and Simpson's rule as separate functions.

ii) Calculate the true integral value for comparison (since the integral of x^2).

Teacher's Signature: Saroj 04/10/24

Experiment → 2.3

Aim : To implement numerical differentiation in Python.

Software Required : Google Colab

Flowchart :

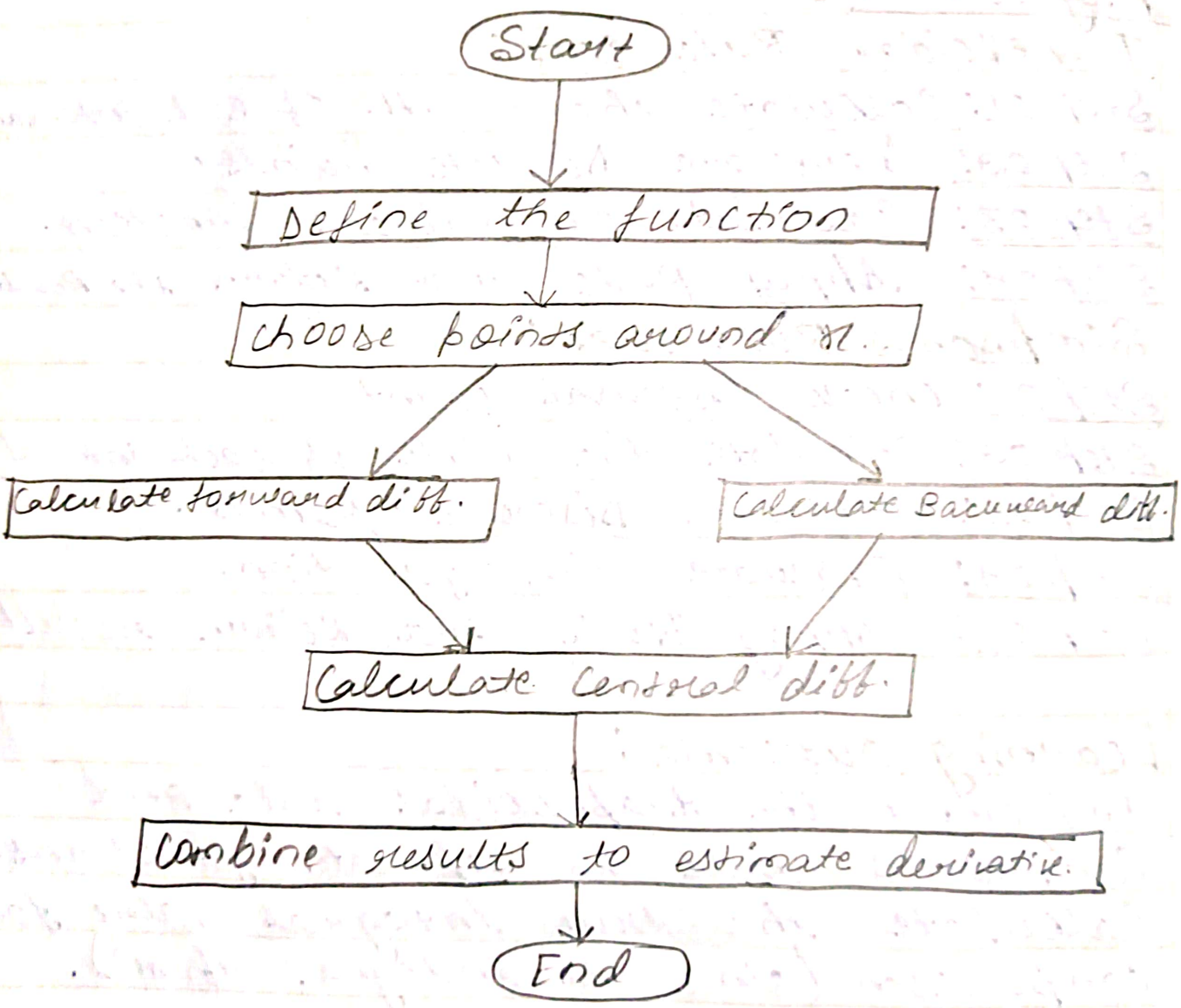


Fig. 1 : Flowchart for implementing a numerical differentiation algorithm.

Date 4/10/24 Expt. No. 9.3

Expt. Name _____ Page No. 19

Aim: To implement numerical differentiation in Python.

Software Required: Google Colab

Description: Numerical differentiation is the process of approximating the derivative of a function using discrete data points, rather than analytic Calculus methods. It is particularly useful when the function is known only at certain points, or when its exact derivative is difficult or impossible to compute analytically. One common approach to numerical differentiation is the finite difference method. This method involves estimating the derivative of a function by using the values of the function at nearby points.

For instance, the forward difference formula can be used to approximate the derivative at a point by calculating the difference between the function value at the point and slightly

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 20

Advanced point.

In python libraries like Numpy provide tools for implementing these methods. By discretizing a given function over a set of intervals, we can calculate the derivative using simple subtraction and division.

Algorithm:

Step 01: Define the function to differentiate.

Step 02: Define the true derivative of the function.

Step 03: Define the finite difference method for numerical differentiation.

- Input:
  function, x, h,
- Output:
  Approximation derivative using forward difference formula:

$$f'(x) \simeq \frac{f(x+h) - f(x)}{h}$$

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 21

Step 04: Set up parameters (x value).

Step 05: Compute the true derivative at x value.

Step 06: Loop through step sizes to compute numerical derivatives & plot tangent lines.

Step 07: Plot the function & tangent lines.

Step 08: Display the graph.

Learning outcomes :

i) Learnt about multiple libraries in python like Numpy & matplotlib.

ii) Learnt about Numerical differentiation techniques.

iii) Learnt about numerical differentiation techniques.

Teacher's Signature: _____

Experiment - 3.1

Aim: To implement procedures of Language multipliers in python.

Software Required: Any python IDE / Python 3.11

Flowchart:

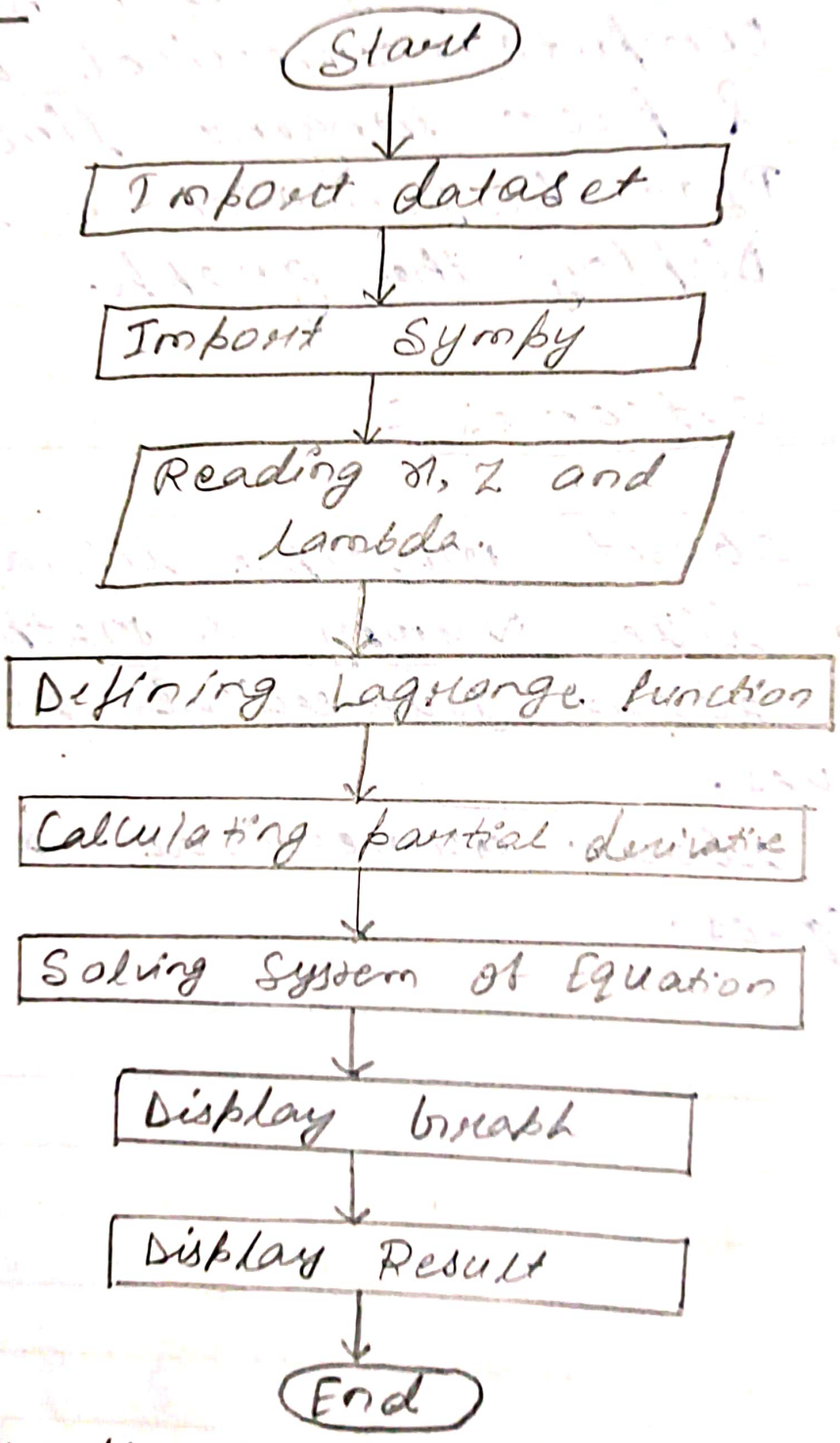


fig.1: Flowchart to display procedures of language multipliers in python.

Date _____ Expt. No. 3.1

Expt. Name _____ Page No. 22

Aim: To implement procedures of Lagrange multipliers in python.

Software Required: Any python IDE | Python 3.11

Description: Lagrange multipliers are a powerful optimization technique for constrained problems. The core idea is to introduce a new variable for each constraint in order to transform a constrained problem into an unconstrained one.

Example :-

Let's minimize the f^n :

$$f(x,y) = x^2 + y^2$$

Subject to the constraint :

$$g(x,y) = x + y - 1 = 0$$

Steps :

i) Define the objective function $f(x,y)$.

ii) Define the constraint $g(x,y) = 0$

iii) construct the Lagrangian :

$$L(x,y,\lambda) = f(x,y) - \lambda(g(x,y))$$

iv) Solve the system of equations formed

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 23

by the partial derivative of $\mathcal{L}$ with respect to x, y, and λ.

Algorithm:

Step 01: Define the variables x, y and λ.
Step 02: Write the objective function $f(x,y)$.
Step 03: Define the constraint $g(x,y) = 0$.
Step 04: Define the Langrangian $\mathcal{L}(x,y,\lambda)$.
Step 05: Take partial derivatives of the Lagrangian w.r.t x, y & λ.
Step 06: Solve the system of equations formed by setting the partial derivatives to zero.

Learning outcomes:

i) Understand the concept of language multipliers and their applications in NLP.
ii) Learnt how to process and manipulate text data for language multipliers.
iii) Analyze the results of language multiplier implementation.

Teacher's Signature

Experiment → 3.2

Aim: Optimization with equality and Inequality constraints using python.

Software Required : Any python IDE | Python 3.11

Flowchart :

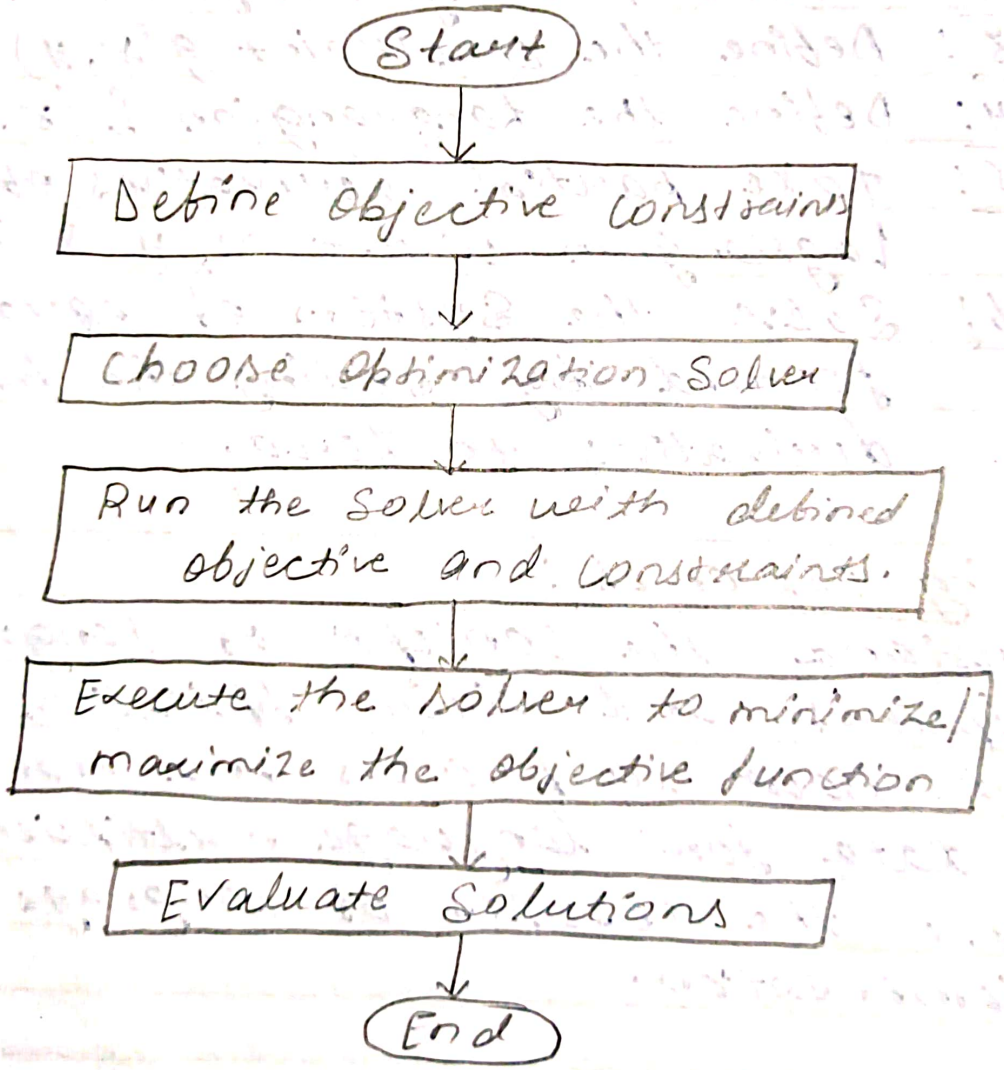


fig.1: Flowchart to displays the process of solving an optimization problem.

Date _____ Expt. No. 3.2

Expt. Name _____ Page No. 24

Aim: Optimization with equality and Inequality constraints using python.

Software Required: Any python IDE, Python 3.11

Description: In Optimization problems, we often encounter constraints on the variables. these constraints can be equality constraints or inequality constraints. the goal is to find a local minimum or maximum of an objective function.

(1) Lagrange multipliers for Equality constraints :- The method extends to problems with equality constraints. The new f' is called Lagrangian, defined as:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

(2) KKT conditions for Inequality constraints :- It generalise Lagrange multiplier to handle both equality & inequal constraints.

The KKT conditions are:

Date _____ Expt. No. _____

Expt. Name _____ Page No. 25

i) Primary feasibility
ii) Dual feasibility
iii) Complementary Slackness

Algorithm:

Step 01: Import Libraries.
Step 02: Define objective function.
Step 03: Define Equality constraint.
Step 04: Define Inequality constraint.
Step 05: Initial guess for variables.
Step 06: Define constraints on dictionary.
Step 07: Constraints Combine into List.
Step 08: Call minimize function as constraints

Learning Outcome:

i) Learnt how to handle different types of constraints during Optimization.
ii) Analyze the feasibility region defined by the constraints.
iii) Explored gradient-based methods.

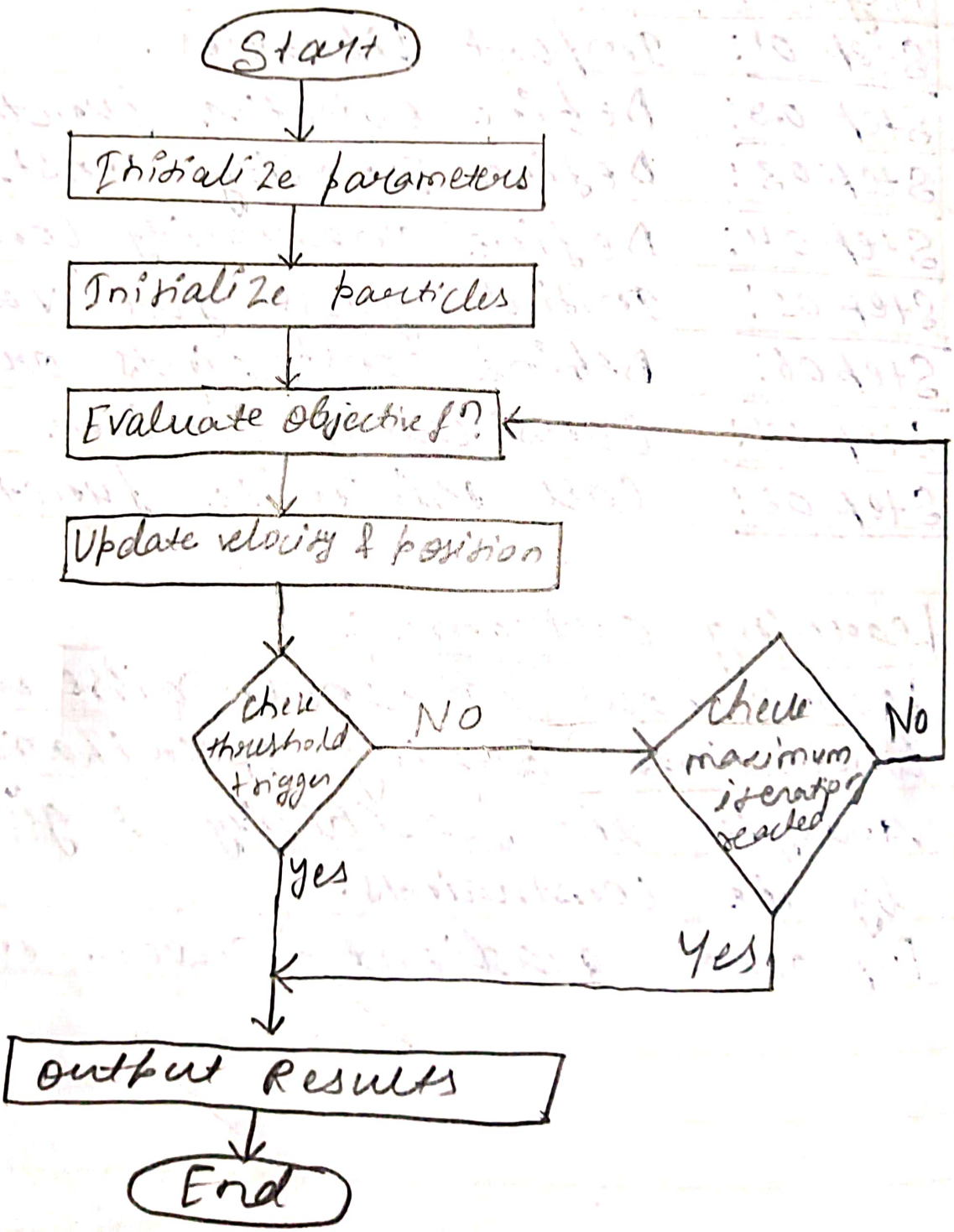
Teacher's Signature:

Experiment → 3.3

Aim: Program to implement triggers using Particle Swarm Optimization (PSO).

Software Required: Python 3.11

Flowchart:



Date _____ Expt. No. 3.3

Expt. Name _____ Page No. 26

Aim: Write a program to optimization of a multidimensional function using particle Swarm optimization to implement triggers.

Software Required: Python 3.11

Description:

The goal of this experiment is to develop a python program that uses Particle Swarm Optimization (PSO) to optimize a multidimensional function. Each particle in the Swarm represents a candidate solution to the optimization problem and moves through the search space by following the current best particles. PSO is particularly useful for optimizing multidimensional functions where traditional gradient-based methods may struggle or fail.

The experiment also aims to implement triggers that signal when certain conditions are met during the

Teacher's Signature: _____

process, such as convergence or reaching a predefined solution threshold.

Algorithm :

Step 01: Define the objective function to be optimized.

Step 02: For each particle, evaluate the objective function at its current position.

Step 03: Update the velocity formula:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i - x_i) + c_2 \cdot r_2 \cdot (g - x_i)$$

where,

$v_i(t)$ = velocity of particle
w = Inertia weight
c_1 & c_2 = Cognitive & Social coefficients
r_1 & r_2 = Random numbers b/w 0 & 1.
p_i = personal best position.
g = global best position.
x_i = Particle's current position.

Step 04: Update particle's position:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Step 05: Check triggers

Step 06: Repeat steps 2-4 until the

Date

Expt. Name

Page No.

stopping criteria are met; i.e; max iterations or convergence.

Step 06: output the global best position and the corresponding value of the objective function.

Learning Outcomes:

i) Learnt basic principles of PSO.
ii) Learnt about various multidimensional functions used in Optimization.
iii) Understand the role of triggers in optimizing algorithms.
iv) Gained skills in analyzing and interpreting the results of PSO.