**National University of Computer & Emerging Sciences**

**Karachi Campus**



**Project Report**

**Numerical Computing Project**

**[]**

**Course Teacher:**

**Muhammad Jamil Usmani**

**Group Members:**

**Zaid Bin Shahab 19K-1512**

**Saud Ahmed Abbasi 19K-0229  
Umer Ahmed 19K-0181**

Contents

[**Sympy Functions Used:** 3](#_Toc73057725)

[**Chapter#2-Solutions of Equations in One Variable** 5](#_Toc73057726)

[Bisection, Regular Falsi, Newton, Secant: 5](#_Toc73057727)

[Fixed Point Iteration: 9](#_Toc73057728)

[**Chapter#3-Interpolation and Polynomial Approximation** 10](#_Toc73057729)

[Lagrange Method : 10](#_Toc73057730)

[Forward Divided Difference: 13](#_Toc73057731)

[Forward Simple Difference: 15](#_Toc73057732)

[Backward Divided Difference: 17](#_Toc73057733)

[Backward Simple Difference: 19](#_Toc73057734)

[Center Divided Difference: 21](#_Toc73057735)

[**Chapter#4-Numerical Differentiation and Integration** 23](#_Toc73057736)

[Three point Formula: 23](#_Toc73057737)

[Five point formulas: 25](#_Toc73057738)

[Trapezoid: 29](#_Toc73057739)

[Simpson’s 1/3: 30](#_Toc73057740)

[Using Simpson’s 3/8 (NO QUESTIONS FOUND IN BOOK, APPLYING ON SIMPSON’S 1/3 QUESTION): 31](#_Toc73057741)

[Midpoint Rule: 32](#_Toc73057742)

[**Chapter#5 -Initial-Value Problems for Ordinary Differential Equations** 34](#_Toc73057743)

[Euler’s Method: 34](#_Toc73057744)

[Modified Euler: 36](#_Toc73057745)

[Huen Method: 38](#_Toc73057746)

[RK4 Method: 40](#_Toc73057747)

# **Sympy Functions Used:**

**Important library functionality imported:**

from sympy.parsing.sympy\_parser import (parse\_expr, standard\_transformations, implicit\_multiplication\_application)

Allows a slightly relaxed syntax.

* Parentheses for single-argument method calls are optional.
* Multiplication is implicit.
* Symbol names can be split (i.e. spaces are not needed between symbols).
* Functions can be exponentiated.

Function parse\_expr() returns an expression of the Type that sympy supports (such as “2x-2” would have the type **sympy.core.add.Add** after passing through the parser), we can also add local\_dict functions that can be detected by the parser (such as sin/cos/tan functions or log functions can be passed directly in the argument of the parser)

For example:  
 Function = parse\_expr(expression, local\_dict={"log": lambda x: sympy.log(x, 10), transformations=(standard\_transformations + (implicit\_multiplication\_application,)))

Where **expression** denotes string input of function, **local\_dict** are the functions we want to have detected while parsing and are replaced by their in-code counterparts, and **transformations** deals with possible functionalities that can be applied such as implicit multiplication (i.e. 2x will be perceived as 2\*x automatically)

All the symbols are detected and stored in a list, here’s an example:  
 all\_symbols = [str(x) for x in **Function**.atoms(Symbol)]

Where **Function** is the function that exists after being string has been parsed by going through the function parse\_expr().

All symbols are replaced with their values, here’s an example:  
 for i, item in enumerate(all\_symbols):

key = all\_symbols[i]

value = limit\_val

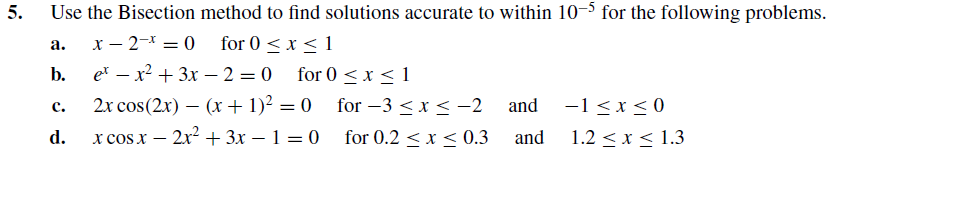
symbol\_vals[key] = value

And then substituted, here’s an example:  
 Function = Function.subs(symbol\_vals)

When all the values have been successfully substituted into the parsed sympy type function, we can now evaluate it using evalf(), and then return the value, here’s an example:  
 return float(Function.evalf())

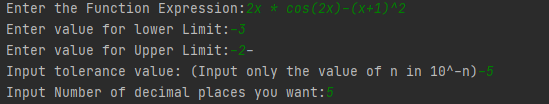
# **Chapter#2-Solutions of Equations in One Variable**

## Bisection, Regular Falsi, Newton, Secant:

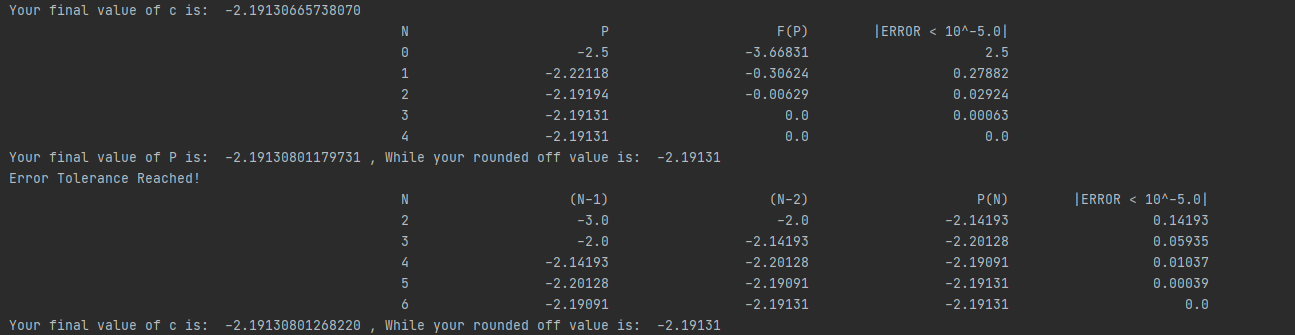
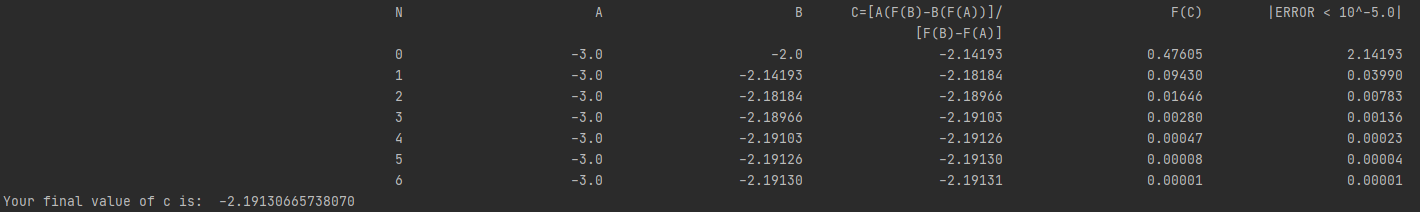
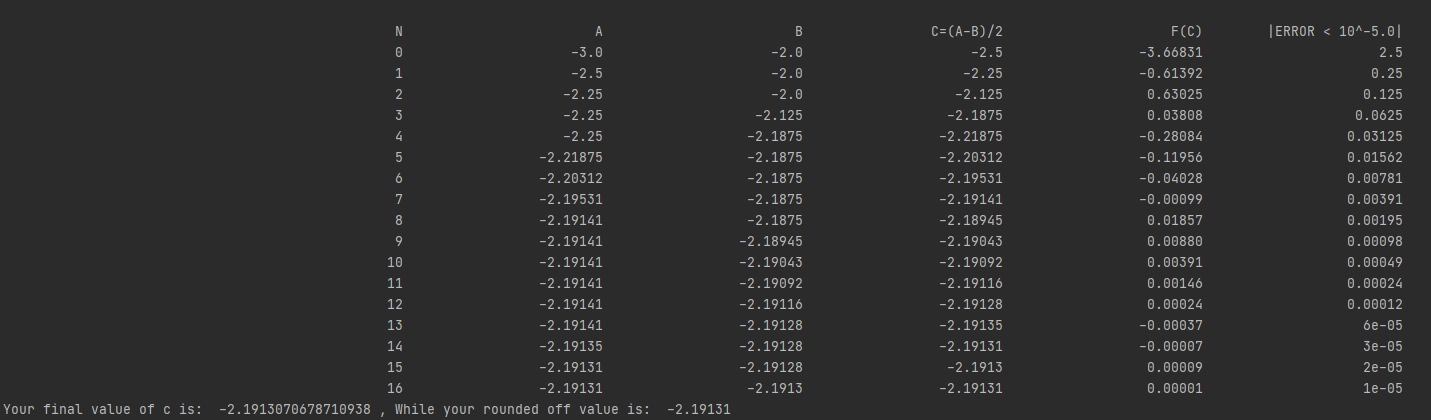


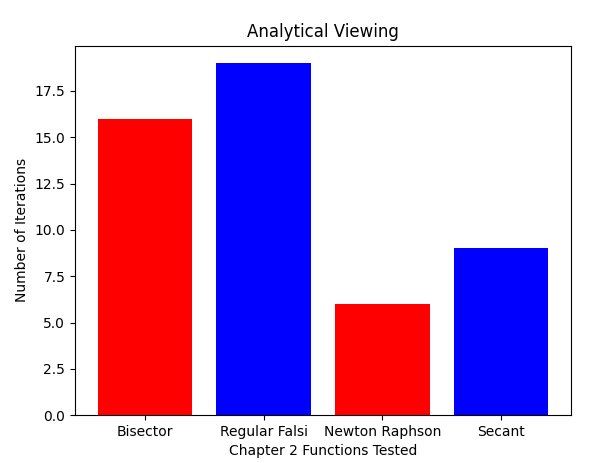
Part Q5c)  
All Methods Applied (i.e. Bisection, Regular Falsi, Newton, Secant)

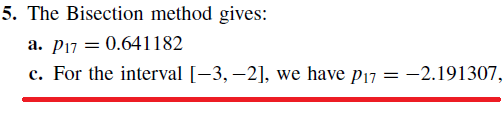
**Input:**



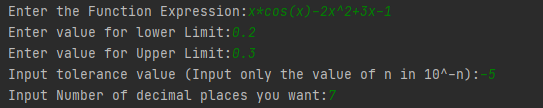
**Output:**

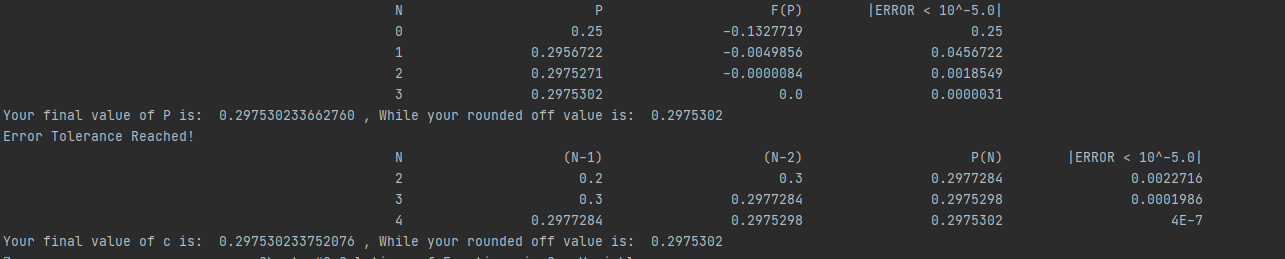
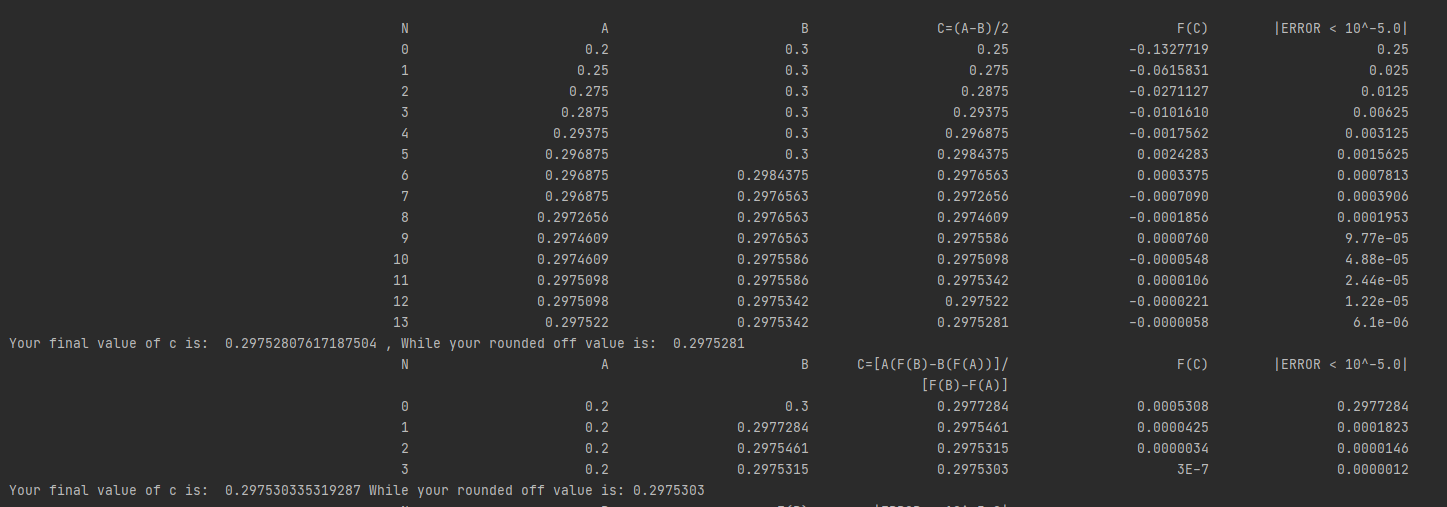


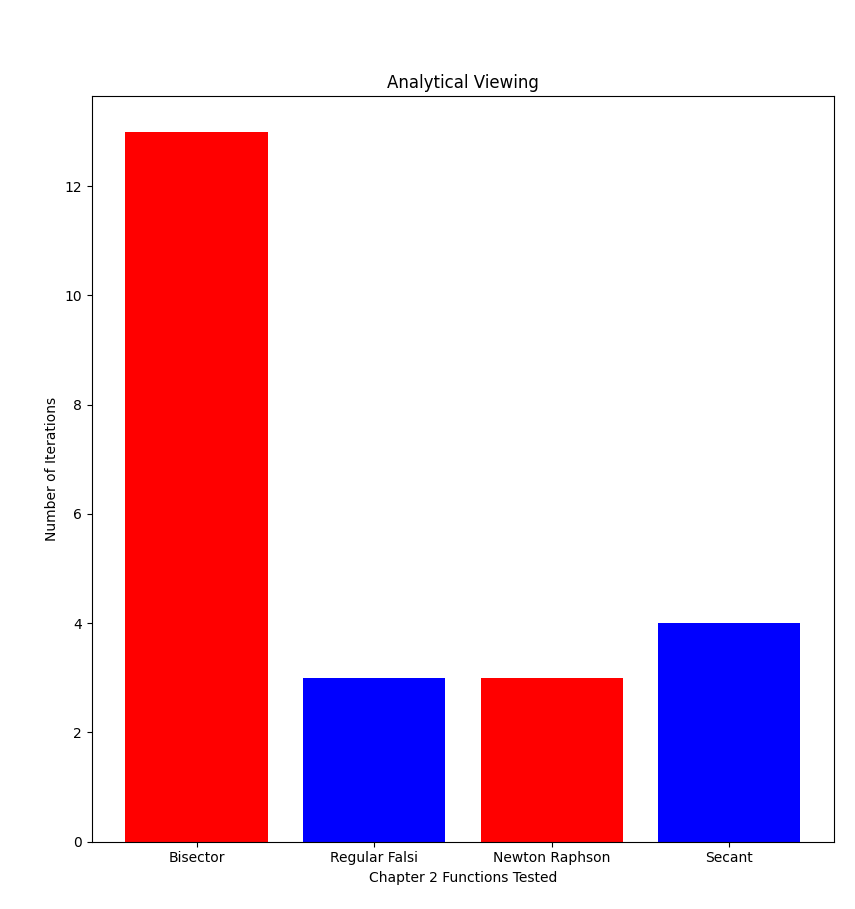


Answer from book matching:  


Q5d)  
All Methods Applied (i.e. Bisection, Regular Falsi, Newton, Secant)

**Input:  
**

**Output:  
**



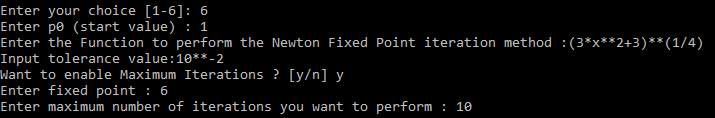
Answer from book matching:



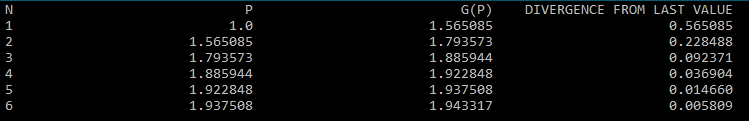
## Fixed Point Iteration:



**Inputs:**

****

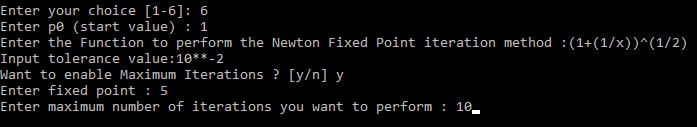
**Outputs:**

****

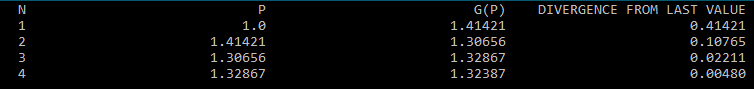
****



**Inputs:**

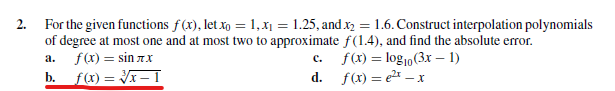
****

**Output:**

****

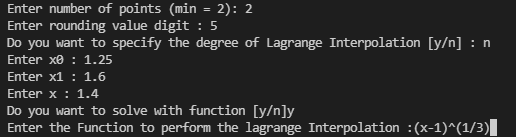
****

# **Chapter#3-Interpolation and Polynomial Approximation**

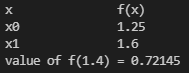
Lagrange Method : ****

**Q2b) using n = 2**

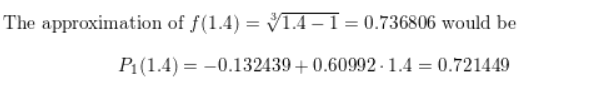
**Input:**

****

**Output:**

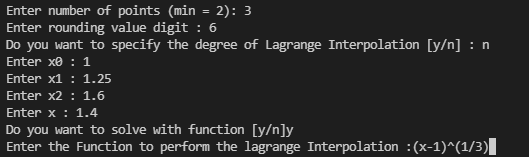
****

**Verification:**

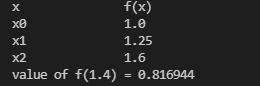
****

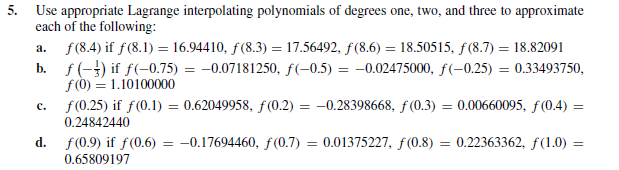
**Q2b) using n = 3**

**Input:**

****

**Output:**

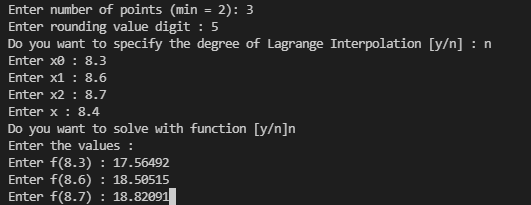
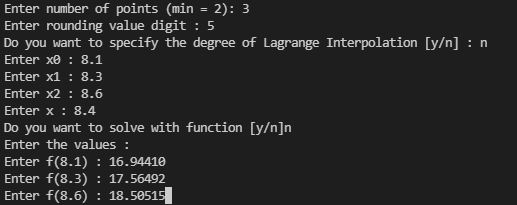
****

****

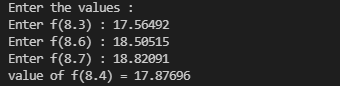
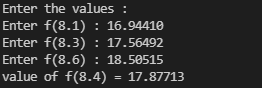
**Q5a) Using Lagrange with n = 3**

**Input:**

**(Upper values)**

**Lower value**

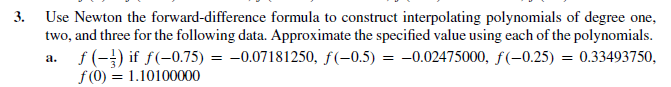
**Output:**

****

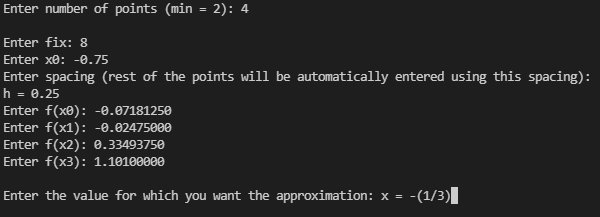
**Verification:**

** from book**

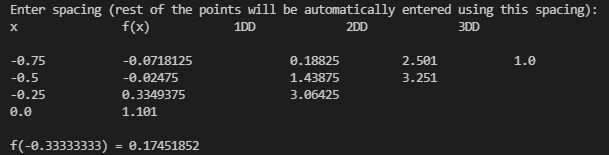
## Forward Divided Difference:

****

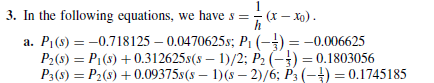
**Input:**

****

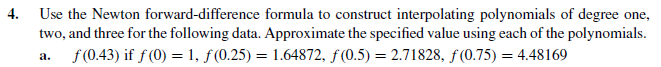
**Output:**

****

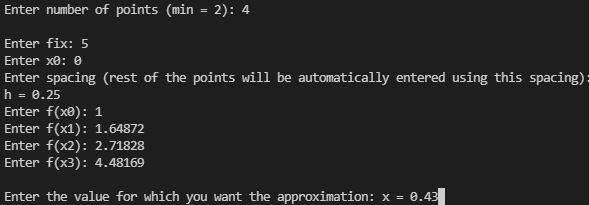
**Verification:**

**🡨-**

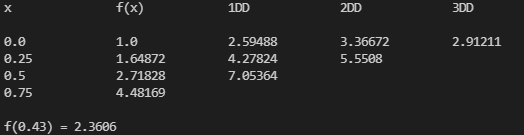
**Another Forward divided difference example:**

****

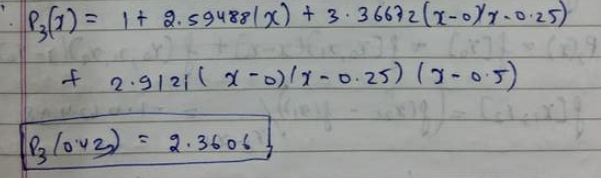
**Input:**

****

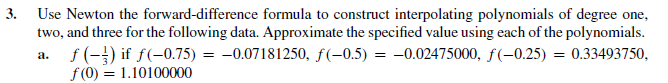
**Output:**

****

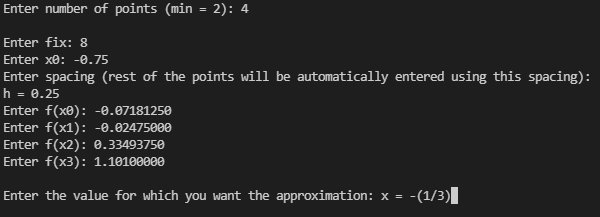
**Verification:**

****

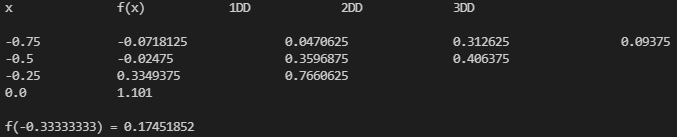
## Forward Simple Difference:

****

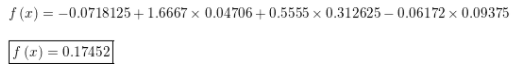
**Input:**

****

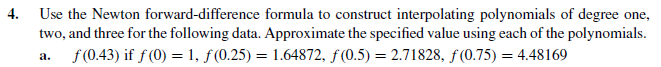
**Output:**

****

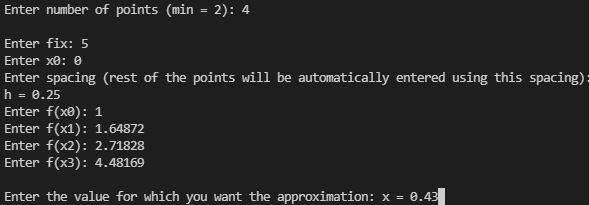
**Verification:**

****

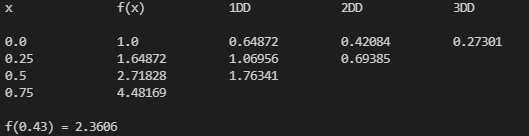
**Another Forward Simple Difference Example:**

****

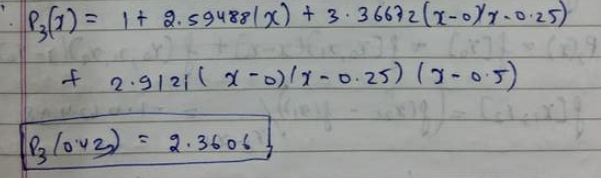
**Input:**

****

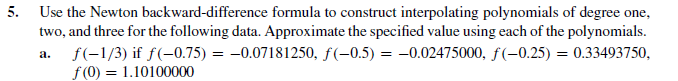
**Output:**

****

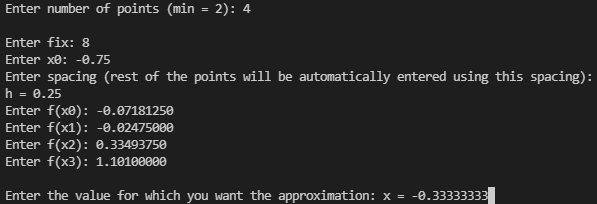
**Verification:**

****

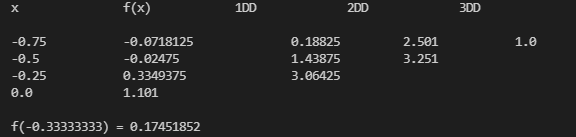
## Backward Divided Difference:

****

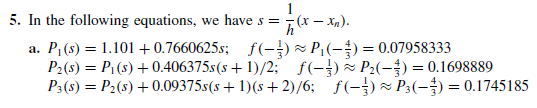
**Input:**

****

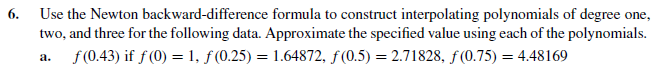
**Output:**

****

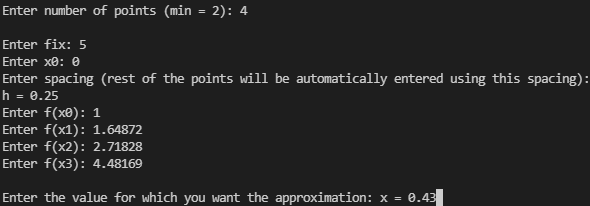
**Verification:**

**🡨**

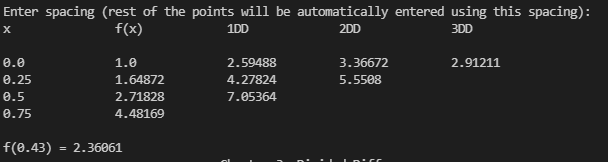
**Another Backward Divided Difference example:**

****

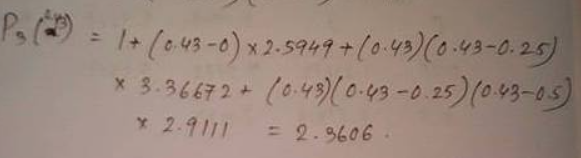
**Input:**

****

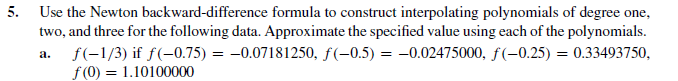
**Output:**

****

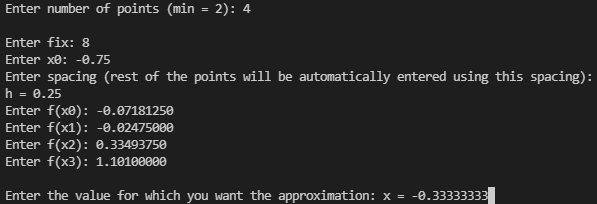
**Verification:**

****

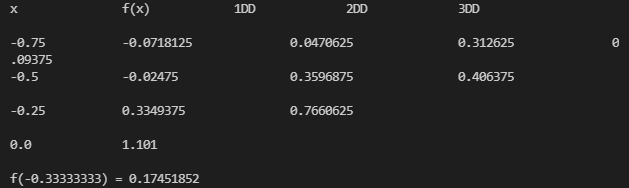
## Backward Simple Difference:

****

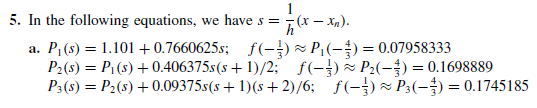
**Input:**

****

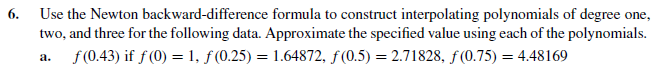
**Output:**

****

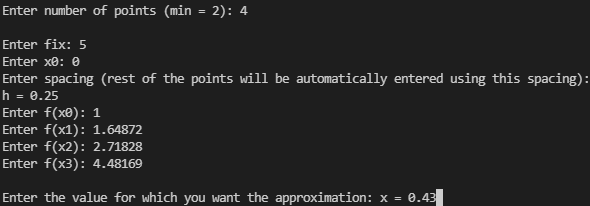
**Verification:**

****

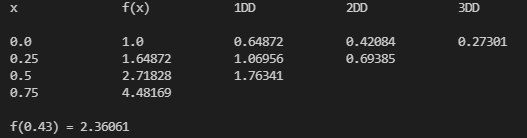
**Another example of Backward Simple Difference:**

****

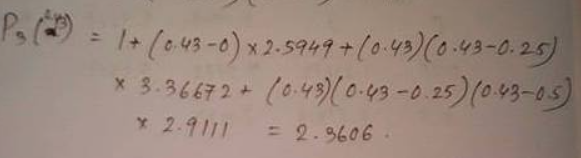
**Input:**

****

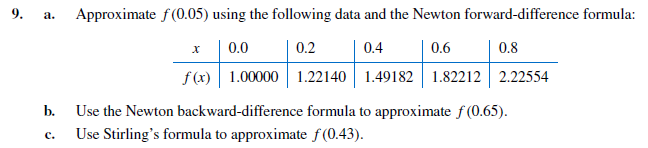
**Output:**

****

**Verification:**

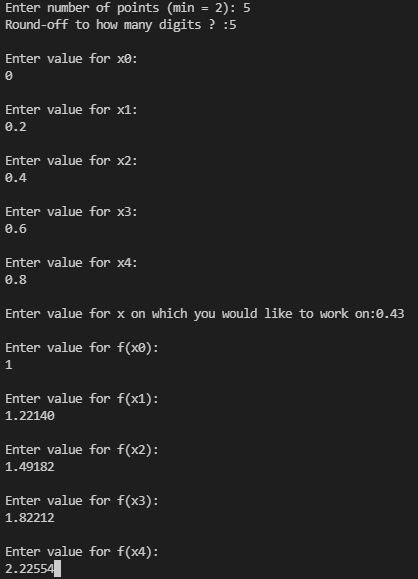
****

## Center Divided Difference:

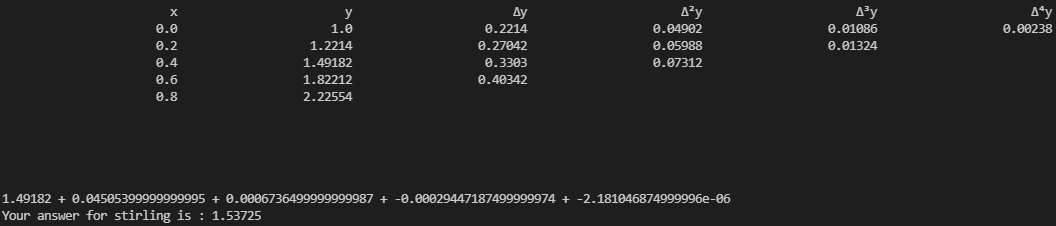
****

**9c)**

**Input:**

****

**Output:**

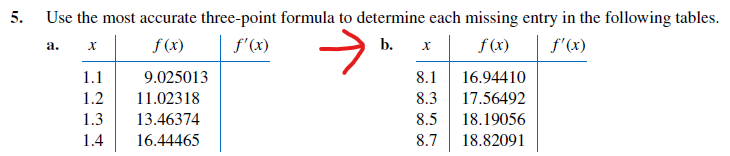
****

**Verification:**

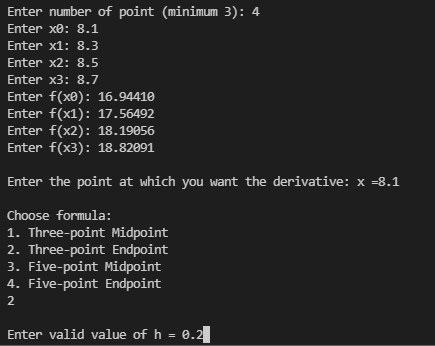
**🡨 from book**

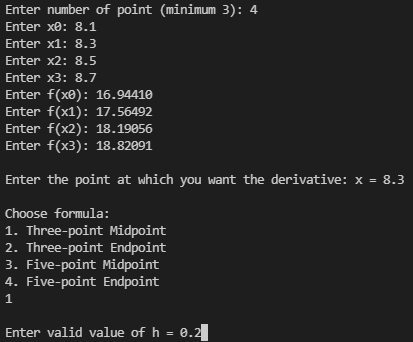
# **Chapter#4-Numerical Differentiation and Integration**

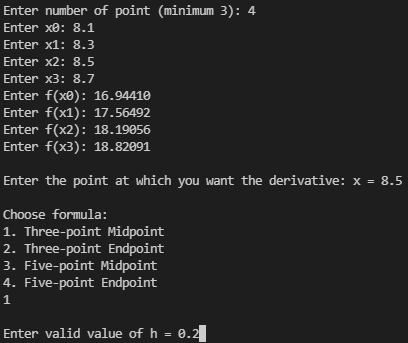
## Three point Formula:

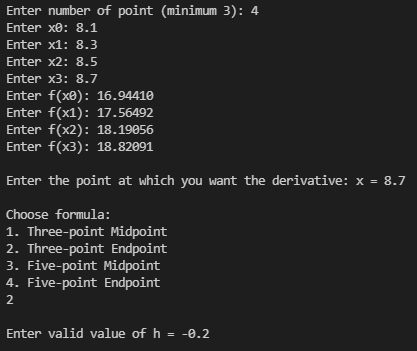
****

**Input:**

****

****

****

****

**Output:**

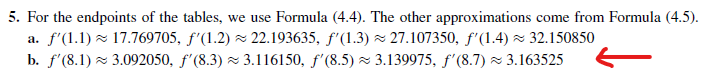
****

****

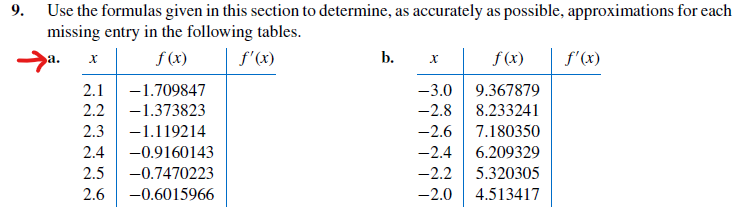
****

****

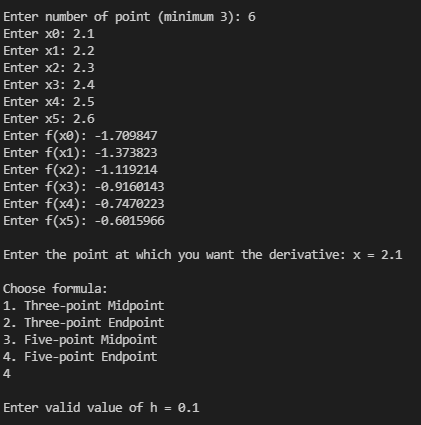
**Verification:**

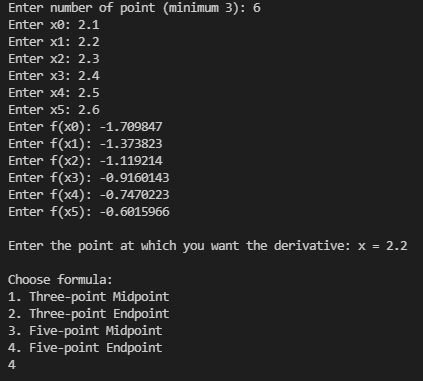
****

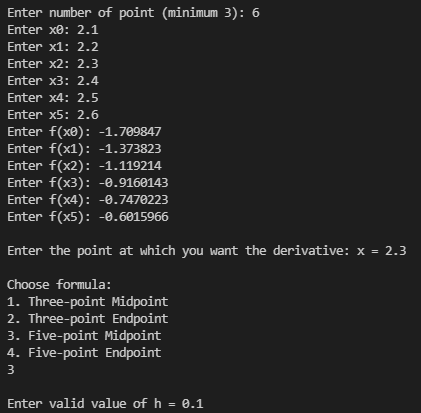
## Five point formulas:

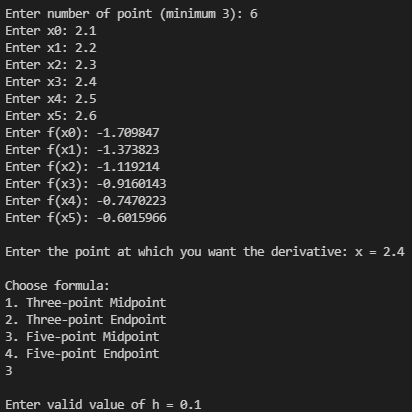
****

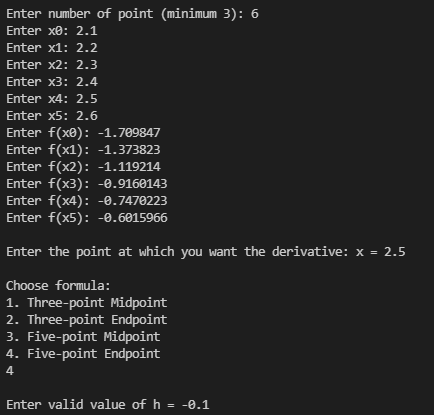
**Input:**

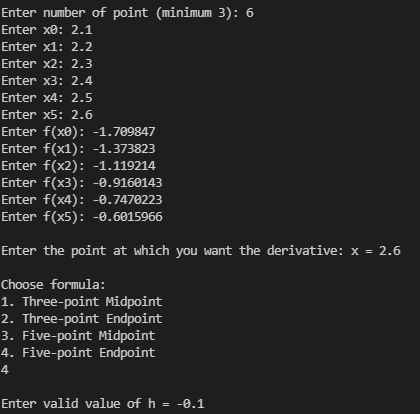
****

****

****

****

****

****

**Output:**

****

****

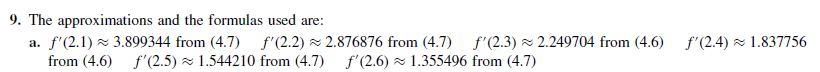
****

****

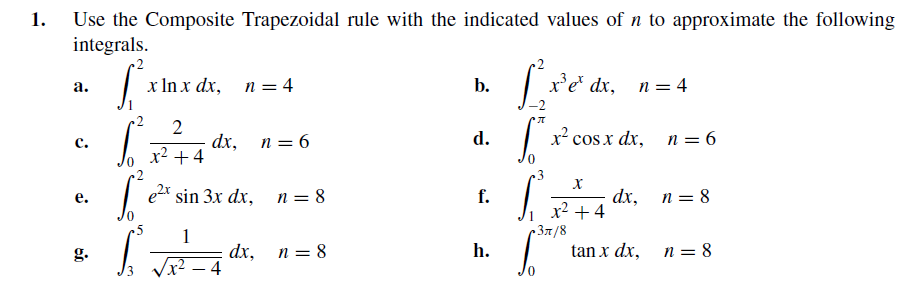
****

****

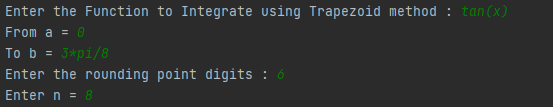
**Verification:**

****

## Trapezoid:



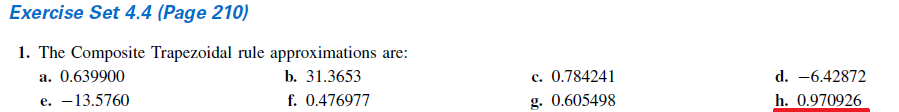
**Q1h)   
Input:**



**Output:**

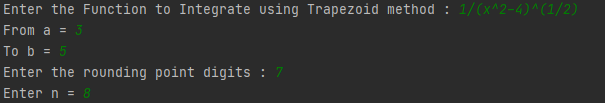


**Verification:**



**Q1g)**

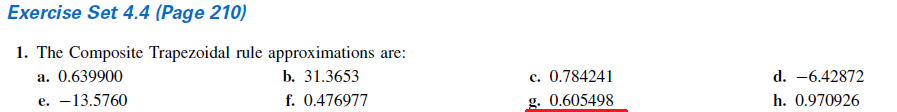
**Input:**

****

**Output:**

****

**Verification:**

****

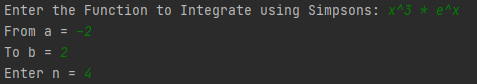
****

## Simpson’s 1/3:

****

**Q3b)**

**Input:**

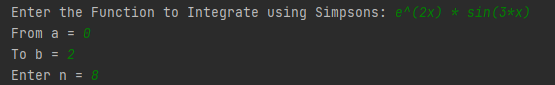
**  
Output:**

****

**Verification:**

**  
Q3e)**

**Input:**

**  
Output:  
**

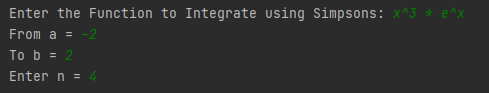
**Verification:**

****

## Using Simpson’s 3/8 (NO QUESTIONS FOUND IN BOOK, APPLYING ON SIMPSON’S 1/3 QUESTION):

**Q3b)**

**Input:**

**  
Output:**

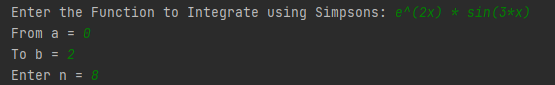
****

**Verification:**

****

**Q3e)**

**Input:**

**  
Output:  
**

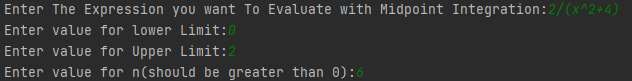
**Verification:**

****

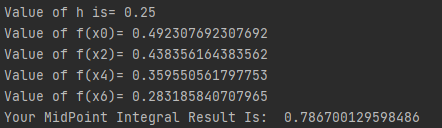
## Midpoint Rule:

**Q5c)**

**Input:**

****

**Output:**

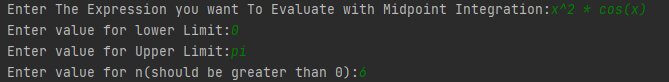
****

**Verification:**

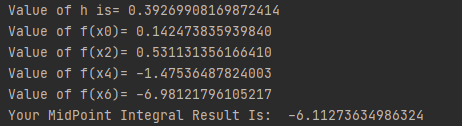
****

**Q5d)**

**Input:**

****

**Output:**

****

**Verification:**

****

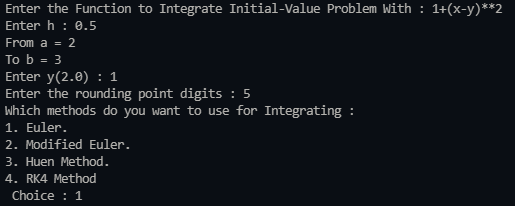
# **Chapter#5 -Initial-Value Problems for Ordinary Differential Equations**

## Euler’s Method:

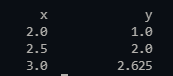
**Ex: 5.2**



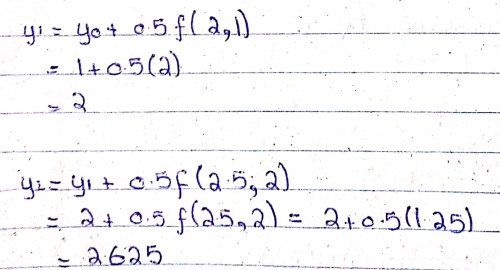
**Inputs:**

****

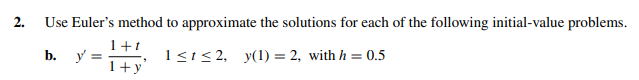
**Output:**

****

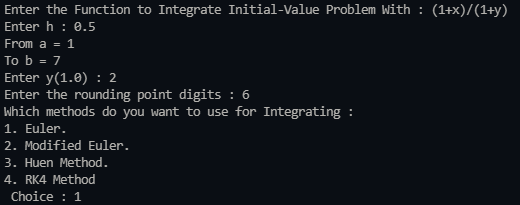
**Verification:**

****

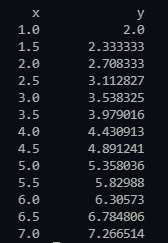
**Ex: 5.2**

****

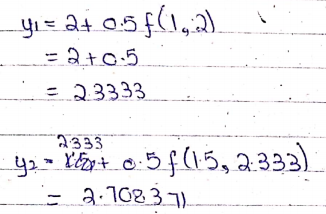
**Inputs:**

****

**Outputs:**

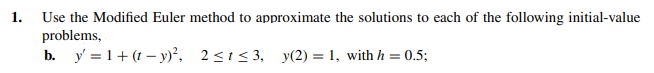
****

**Verification:**

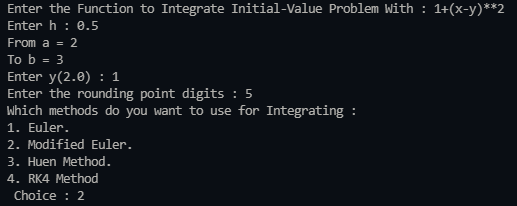
****

## Modified Euler:

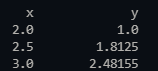
**Ex: 5.4**



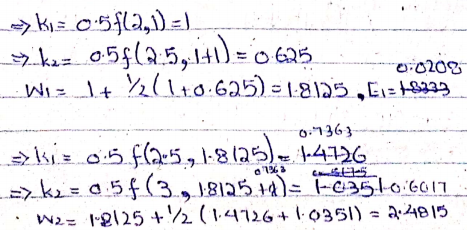
**Inputs:**

****

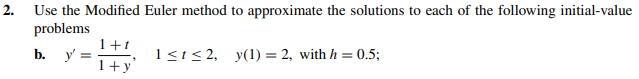
**Output:**

****

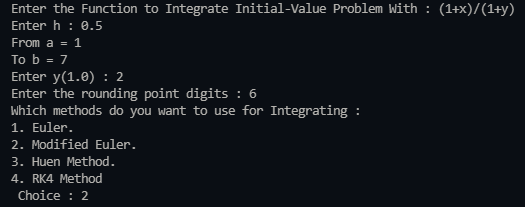
**Verification:**

****

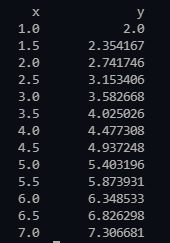
**Ex: 5.4**

****

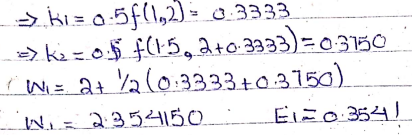
**Inputs:**

****

**Outputs:**

****

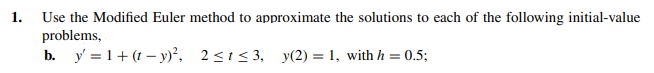
**Verification:**

****

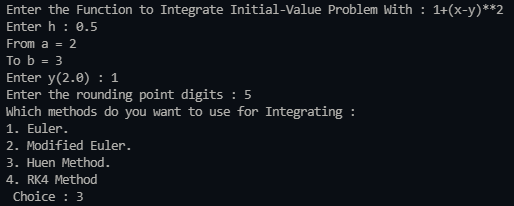
## Huen Method:

**Ex: 5.4**

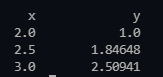
****



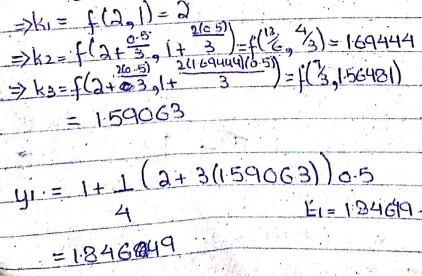
**Input:**

****

**Output:**

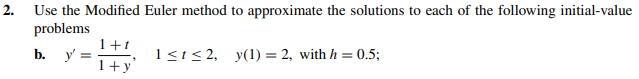
****

**Verification:**

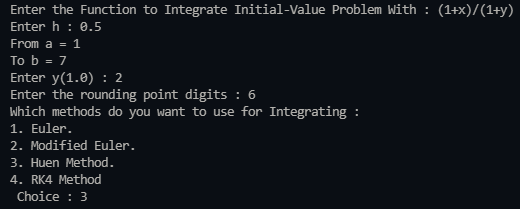
****

**Ex: 5.4**

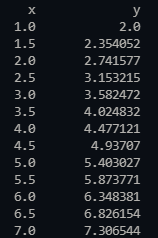
****

****

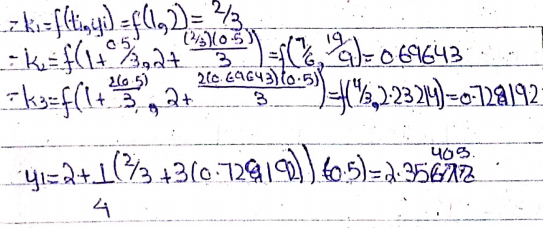
**Inputs:**

****

**Output:**

****

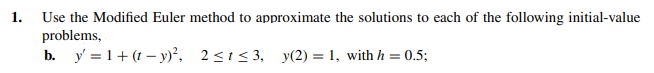
**Verification:**

****

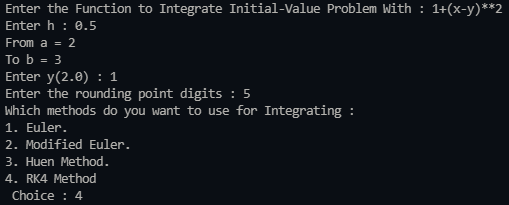
## RK4 Method:

**Ex: 5.4**

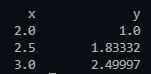
****



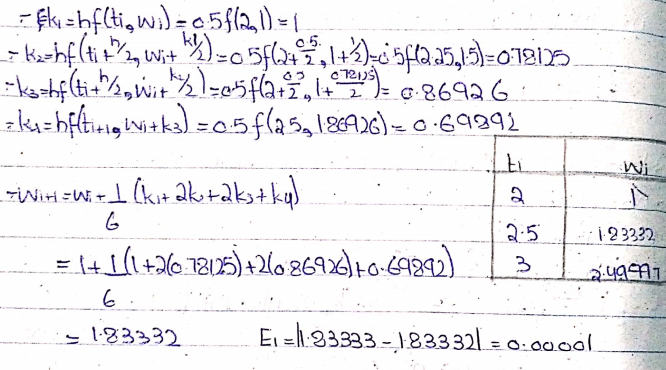
**Inputs:**

****

**Output:**

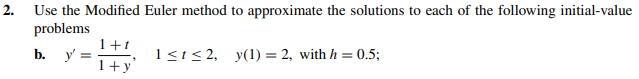
****

**Verification:**

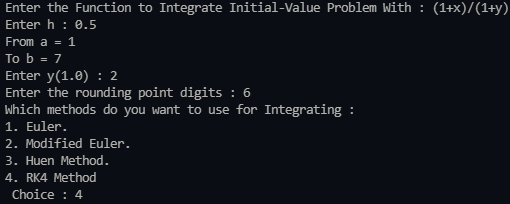
****

**Ex: 5.4**

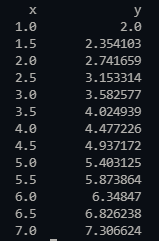
****

****

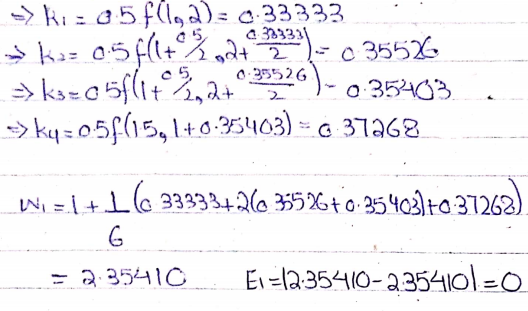
**Inputs:**

****

**Output:**

****

**Verification:**

****