National University of Computer and Emerging Sciences



Name: Muhammad Suleman

Roll #: 22F-3350

Section: BCS-4B

Assignment # 01-Part01

# Question 01

## Newton Raphson

## Code

*import* sympy

*import* matplotlib.pyplot *as* plt

x\_values = []

y\_values = []

*def* newton\_raphson(f, df, x0, tol=1e-6, max\_iter=100):

*"""*

*Newton-Raphson method for finding the root of a function.*

*Args:*

*f (function): The function for which we want to find the root.*

*df (function): The derivative of the function.*

*x0 (float): Initial guess for the root.*

*tol (float, optional): Tolerance for convergence. Defaults to 1e-6.*

*max\_iter (int, optional): Maximum number of iterations. Defaults to 100.*

*Returns:*

*float: The approximate root of the function.*

*"""*

x\_values.append(x0)

x = x0

print("\n----------- Iteration Table -----------")

print("n\t\t x(n)\t\t\tx(n+1)")

print("---------------------------------------")

*for* i *in* range(max\_iter):

x\_new = x - f(x) / df(x)

y\_values.append(x\_new)

print(i, "\t\t", f"{x:.6f}\t\t{x\_new:.6f}")

*if* abs(x\_new - x) < tol:

*return* x\_new

x = x\_new

x\_values.append(x)

*raise* ValueError("Newton-Raphson method did not converge.")

print(">>>>>>>>>>>> Newton Raphson Method <<<<<<<<<<<<")

*# main()*

*# Example usage: user input for equation and initial guess*

equation = input("Enter your equation (in terms of x): ")

initial\_guess = float(input("Enter an initial guess for the root: "))

*# Define the function and its derivative based on user input*

x = sympy.symbols('x')

expr = sympy.sympify(equation)

f = sympy.lambdify(x, expr)

*# Calculate the derivative symbolically*

df = sympy.lambdify(x, sympy.diff(expr, x))

*# Find the root using Newton-Raphson method*

root = newton\_raphson(f, df, initial\_guess)

print(f"Approximate root: {root:.6f}")

*# Plotting*

plt.plot(x\_values, y\_values)

plt.xlabel('x')

plt.ylabel('x(n+1)')

plt.title('Newton Raphson Method')

plt.grid(*True*)

plt.show()

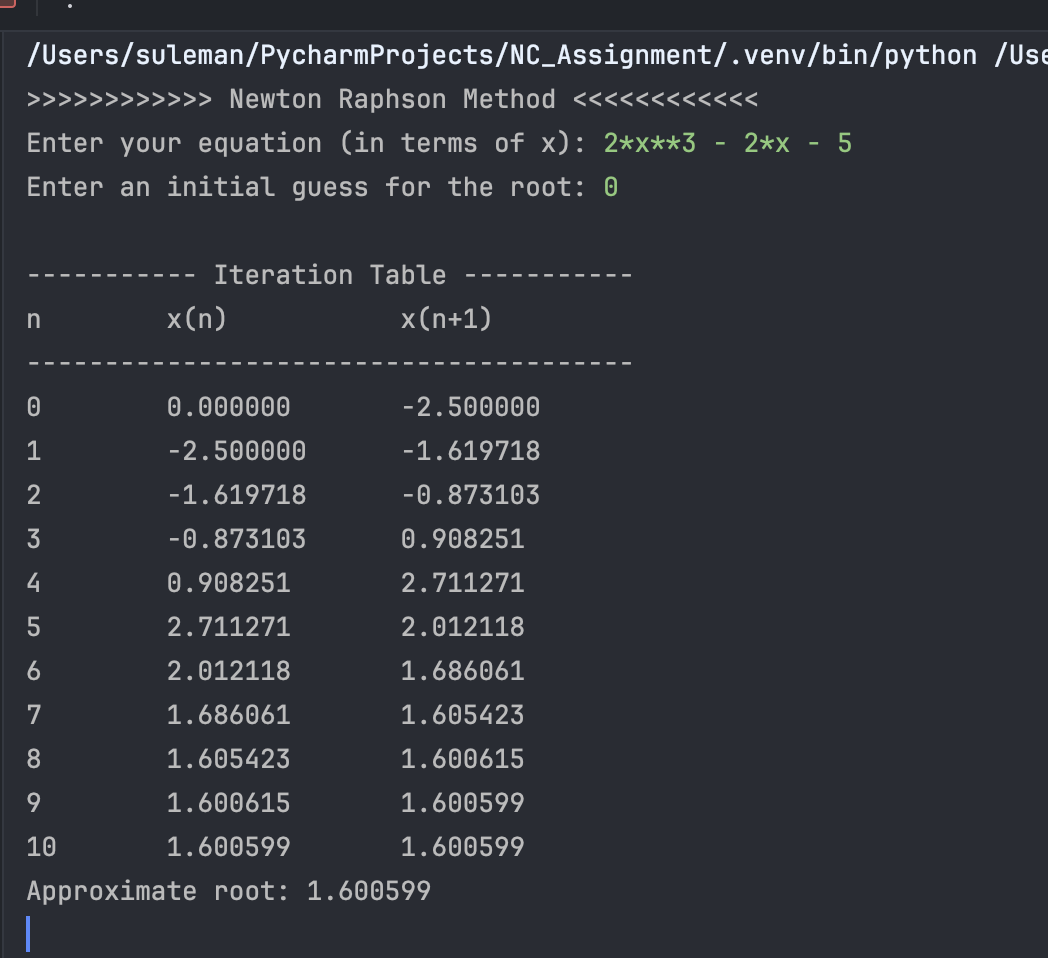
*# Q1 2\*x\*\*3 - 2\*x - 5 , initial guess = 0*

*# Q2 2\*\*x - x - 1.7 , initial guess = 0*

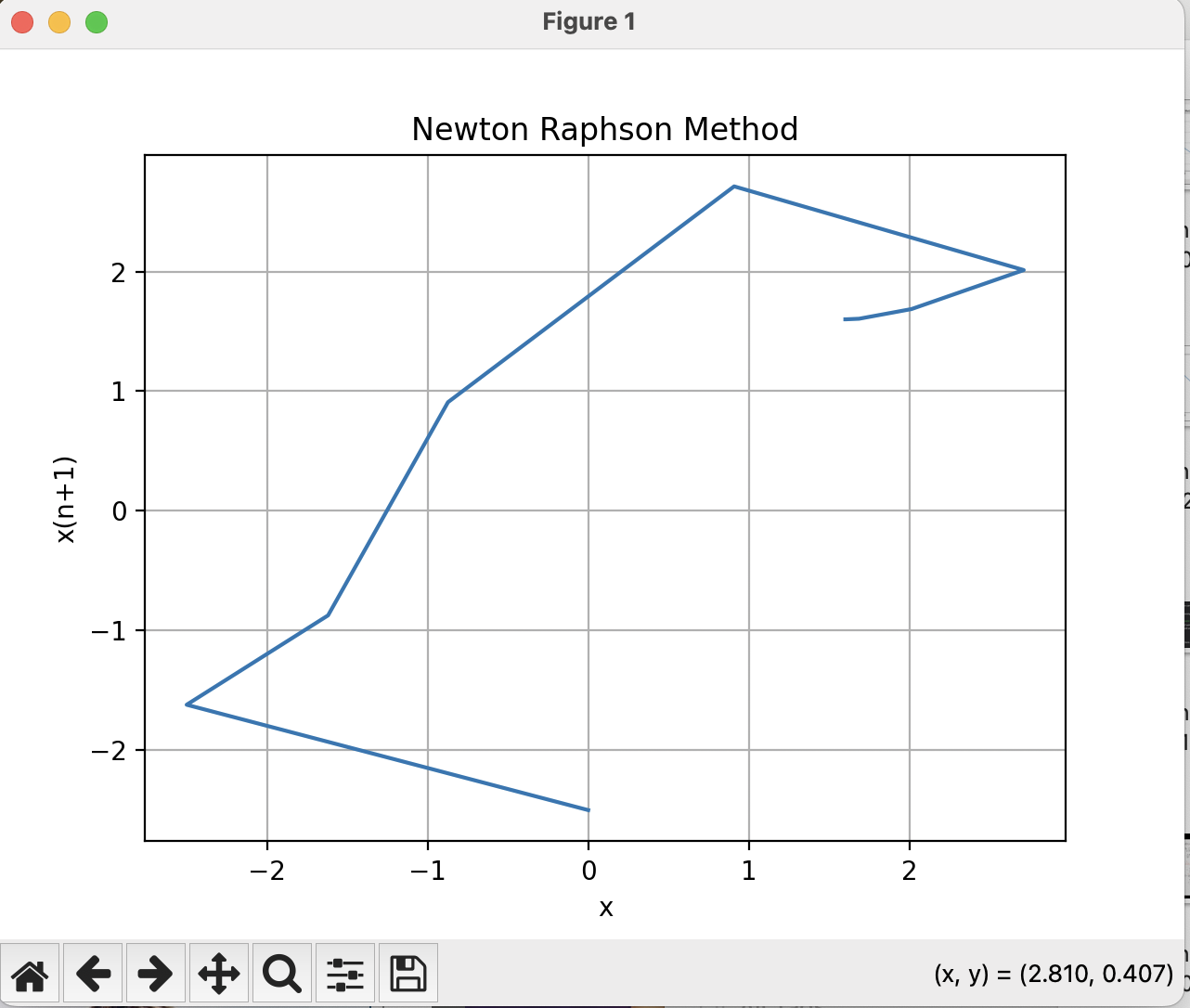
### Sample Problem 01

### 2\*x\*\*3 - 2\*x - 5 , initial guess = 0

### Output



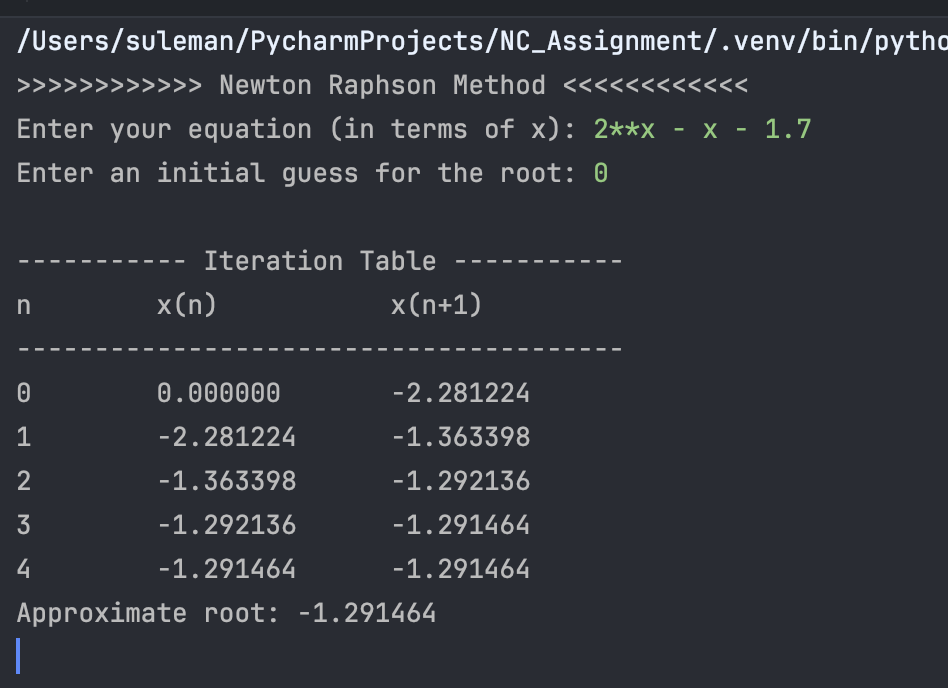
### Graph Plot



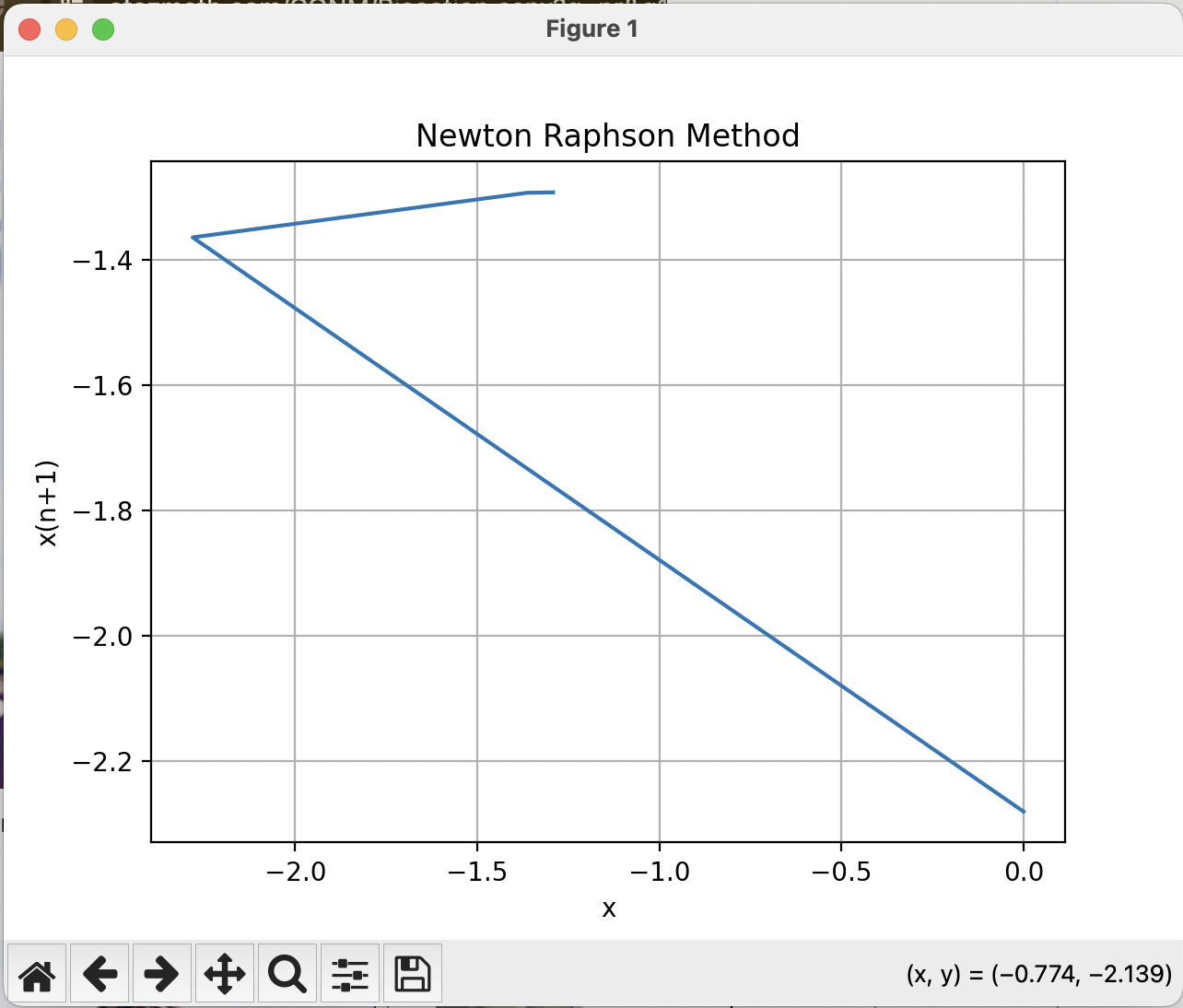
### Sample Problem 02

2\*\*x - x - 1.7 , initial guess = 0

### Output



### Graph Plot



# Question 02-Part(iii)

## Euler Method

## Code

*import* matplotlib.pyplot *as* plt

x\_values = []

y\_values = []

*def* euler\_method(f, x0, y0, h, xn):

*"""*

*Solves the ODE dy/dx = f(x, y) using the Explicit Euler method.*

*Args:*

*f: A function that computes the derivative dy/dx.*

*x0: Initial value of x.*

*y0: Initial value of y.*

*h: Step size.*

*xn: Calculation point.*

*Returns:*

*Arrays containing x and y values.*

*"""*

print("\n----------- Iteration Table -----------")

print("n\t x(n)\t\ty(n)\t\ty(n+1)")

print("---------------------------------------")

x\_values.append(x0)

y\_values.append(y0)

i = 0

*while* x\_values[-1] < xn:

x\_new = x\_values[-1] + h

y\_new = y\_values[-1] + h \* f(x\_values[-1], y\_values[-1])

x\_values.append(x\_new)

y\_values.append(y\_new)

print(i, "\t", f"{x0:.6f}\t{y0:.6f}\t{y\_new:.6f}")

i += 1

y0 = y\_new

x0 += h

*return* x\_values, y\_values

print(">>>>>>>>>>>> Euler Method <<<<<<<<<<<<")

*# main()*

*# Prompt the user for function input*

function\_str = input("Enter the function f(x, y): ")

my\_function = eval("lambda x, y: " + function\_str) *# Convert string to function*

*# Prompt the user for initial values*

x0 = float(input("Enter the initial value of x: "))

y0 = float(input("Enter the initial value of y: "))

*# Prompt the user for step size*

step\_size = float(input("Enter the step size (h): "))

xn = float(input("Enter calculation point (xn): "))

*# Compute using Euler Method*

x\_values, y\_values = euler\_method(my\_function, x0, y0, step\_size, xn)

*# Print the results with six decimal places*

print("\nAt x = {:.6f}, y = {:.6f}".format(x\_values[-1], y\_values[-1]))

*# Ploting the lists X-Values and Y-Values*

plt.plot(x\_values, y\_values)

plt.xlabel('x')

plt.ylabel('y')

plt.title('Euler Method')

plt.grid(*True*)

plt.show()

*# function f(x, y): -(2\*y)/x + 2/(3\*x) - (4\*(1)\*(x))/3*

*# initial value of x: 0.1*

*# initial value of y: 0.1*

*# step size (h): 0.7*

*# calculation point (xn): 5*

*# function f(x, y): -(x \* y\*\*2 + y)*

*# initial value of x: 0*

*# initial value of y: 1*

*# step size (h): 0.1*

*# calculation point (xn): 1*

### Sample Problem 01

f(x, y): -(2\*y)/x + 2/(3\*x) - (4\*(1)\*(x))/3

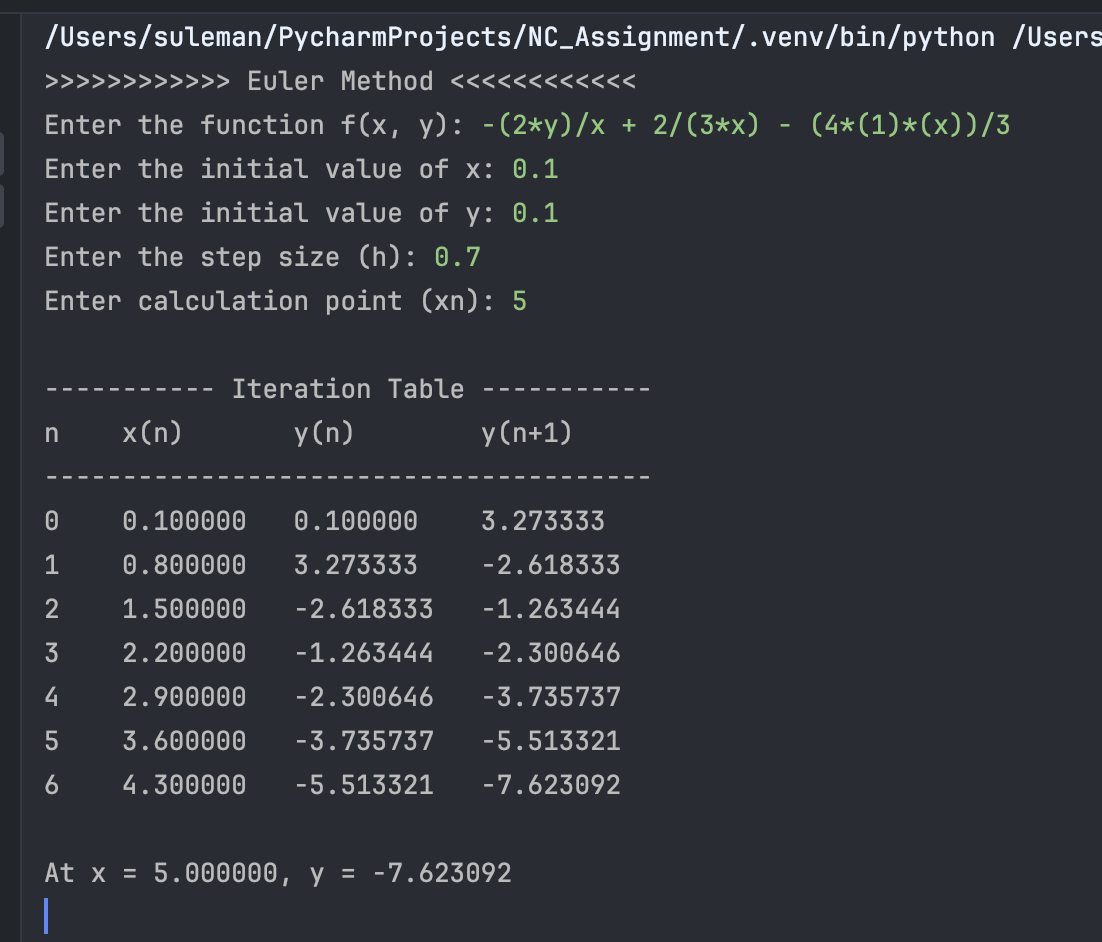
initial value of x: 0.1

initial value of y: 0.1

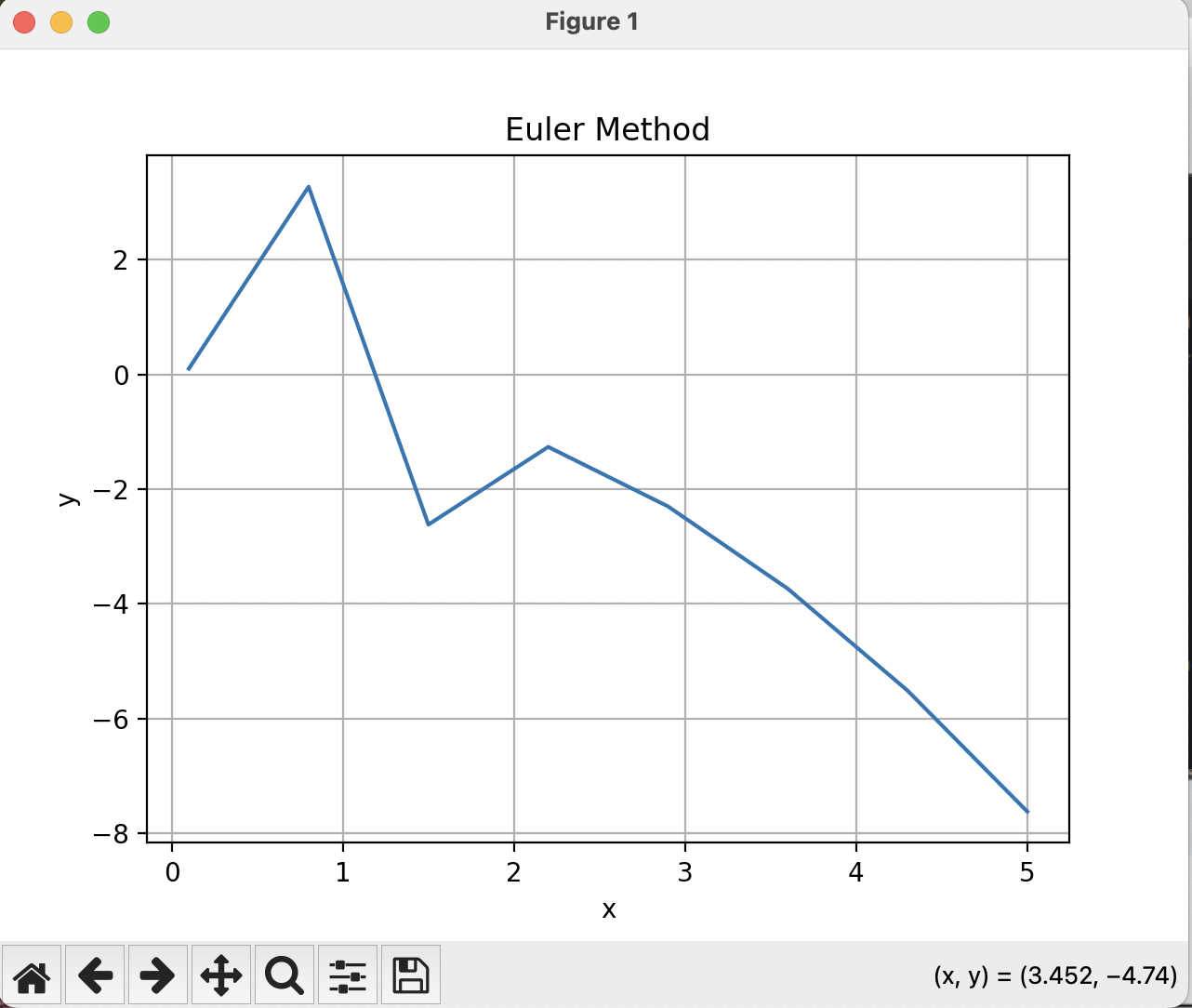
step size (h): 0.7

calculation point (xn): 5

### Output



### Graph Plot



### Sample Problem 02

function f(x, y): -(x \* y\*\*2 + y)

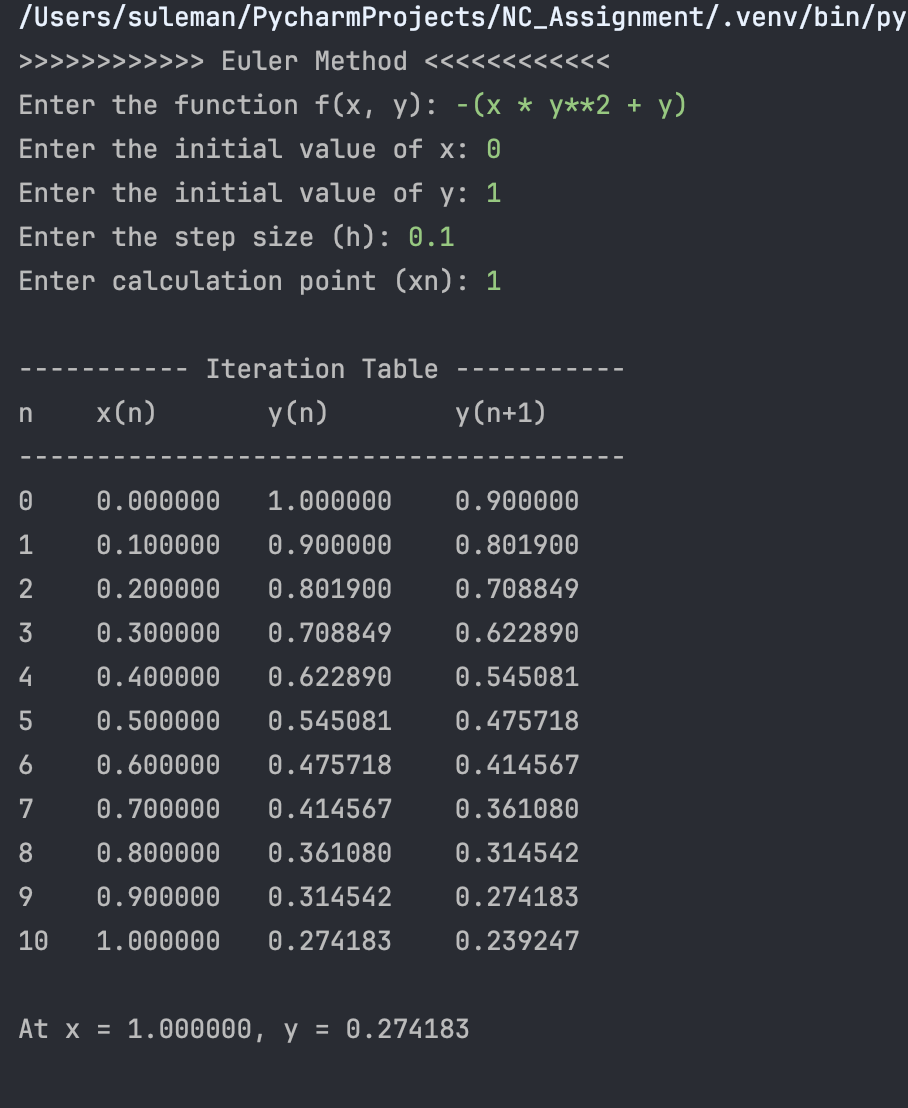
initial value of x: 0

initial value of y: 1

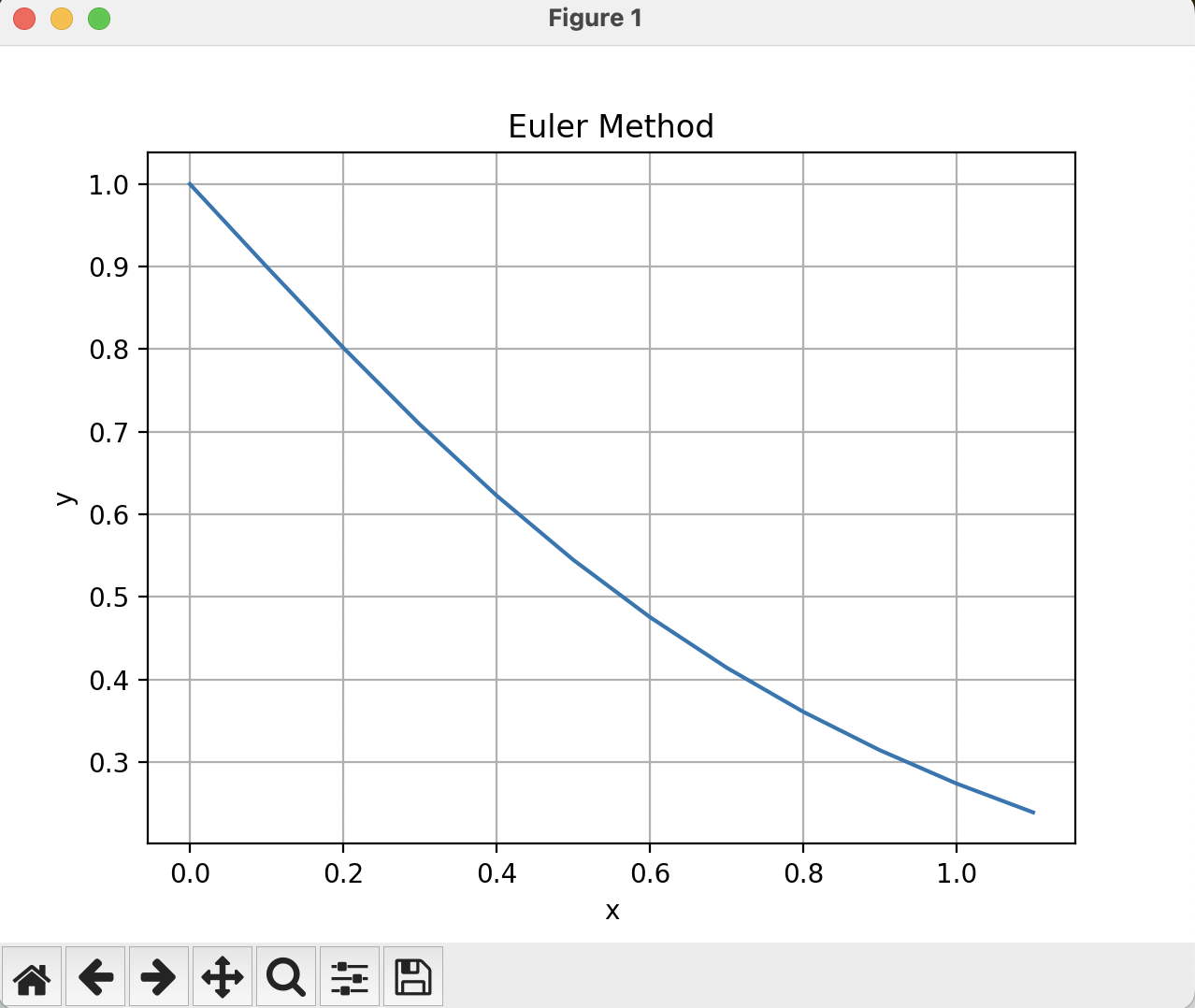
step size (h): 0.1

calculation point (xn): 1

### Output



### Graph Plot



## RK Method of Order 4

## Code

*import* matplotlib.pyplot *as* plt

*# Created empty list to keep track of values for ploting*

x\_values = []

y\_values = []

*def* f(x, y):

*return* eval(equation) *# Evaluate the user-provided equation*

*def* rk4(x0, y0, xn, h):

*"""*

*Runge-Kutta Fourth Order method implementation.*

*Args:*

*x0 (float): Initial value of x.*

*y0 (float): Initial value of y.*

*xn (float): Calculation point where y is evaluated.*

*h (float): Step size.*

*"""*

print("\n----------- Iteration Table -----------")

print("n\t x(n)\t\ty(n)\t\ty(n+1)")

print("---------------------------------------")

*#n (int): Number of steps(Iteration).*

n = (int)((xn - x0) / h)

x\_values.append(x0)

y\_values.append(y0)

i = 0

*for* i *in* range(n):

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

yn = y0 + k

y\_values.append(yn)

print(i, "\t", f"{x0:.6f}\t{y0:.6f}\t{yn:.6f}")

i += 1

y0 = yn

x0 += h

x\_values.append(x0)

print("\nAt x = {:.6f}, y = {:.6f}".format(xn, yn))

print(">>>>>>>>>>>> RK Method of Order 4 <<<<<<<<<<<<")

*# main()*

*# User inputs*

equation = input("Enter the f(x,y): ")

x0 = float(input("Enter initial value of x (x0): "))

y0 = float(input("Enter initial value of y (y0): "))

xn = float(input("Enter calculation point (xn): "))

step\_size = float(input("Enter step size (h): "))

*# Call RK4 method*

rk4(x0, y0, xn, step\_size)

plt.plot(x\_values,y\_values)

plt.xlabel('x')

plt.ylabel('y')

plt.title('RK Method of Order 4')

plt.grid(*True*)

plt.show()

*# f(x, y): -(2\*y)/x + 2/(3\*x) - (4\*(1)\*(x))/3*

*# initial value of x: 0.1*

*# initial value of y: 0.1*

*# step size (h): 0.7*

*# calculation point (xn): 5*

*# function f(x, y): -(x \* y\*\*2 + y)*

*# initial value of x: 0*

*# initial value of y: 1*

*# step size (h): 0.1*

*# calculation point (xn): 1*

### Sample Problem 01

f(x, y): -(2\*y)/x + 2/(3\*x) - (4\*(1)\*(x))/3

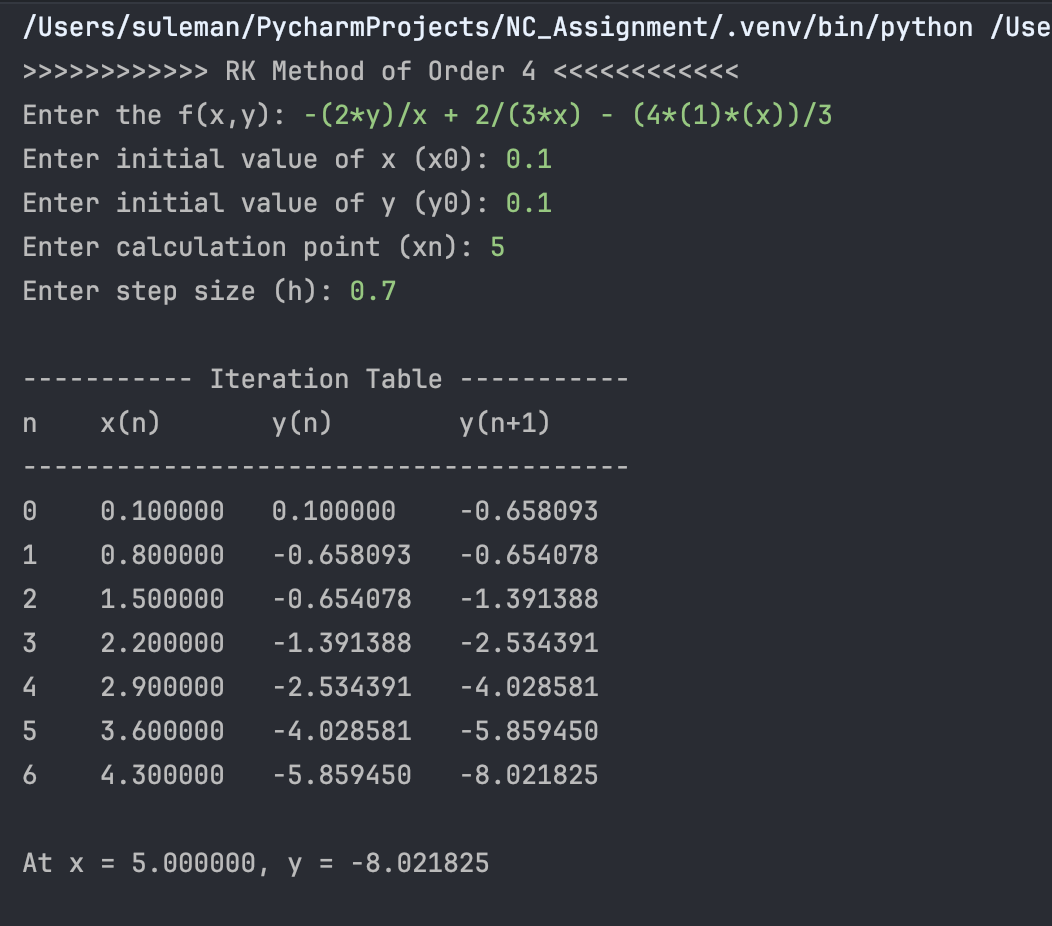
initial value of x: 0.1

initial value of y: 0.1

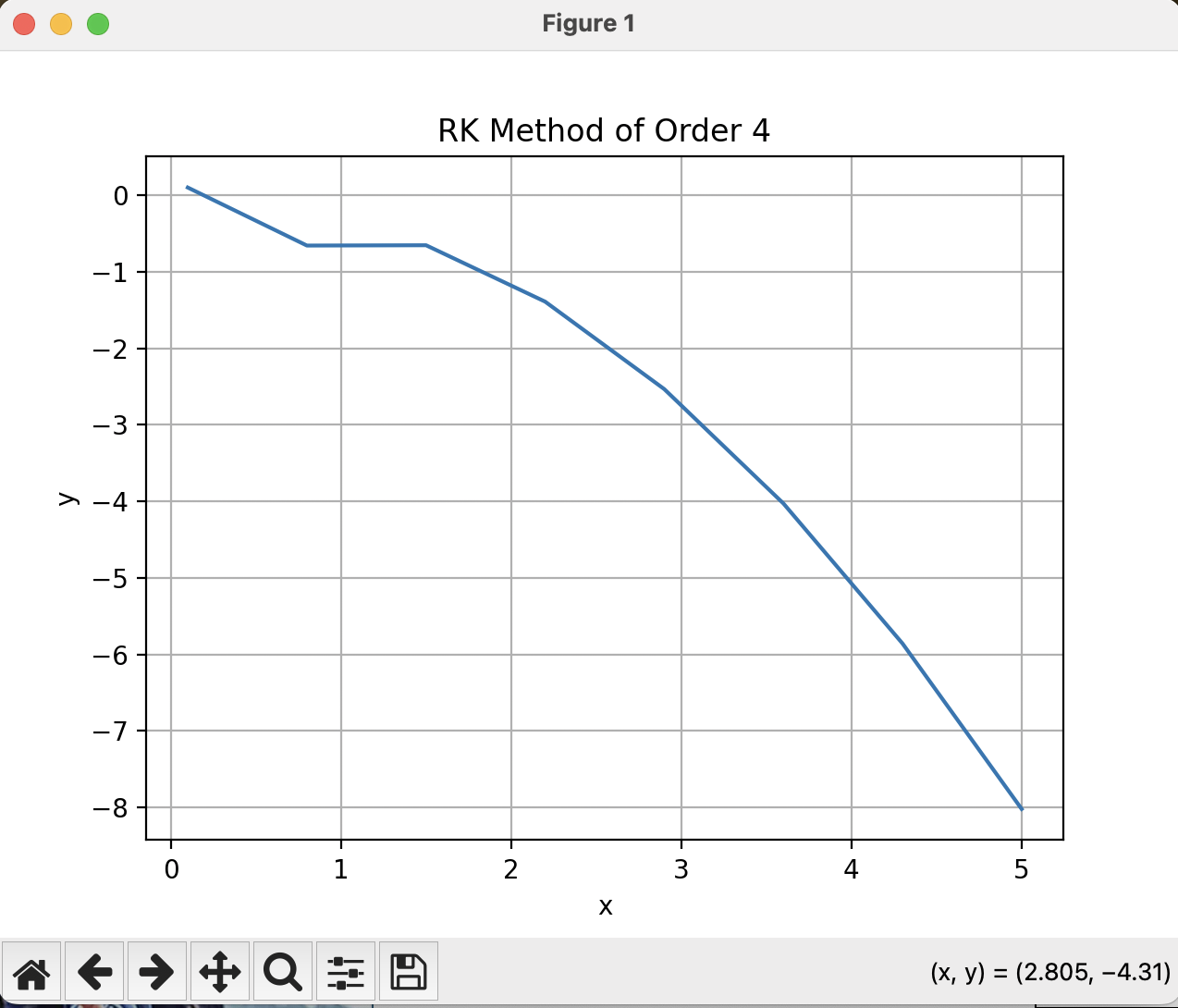
step size (h): 0.7

calculation point (xn): 5

### Output



### Graph Plot



### Sample Problem 02

function f(x, y): -(x \* y\*\*2 + y)

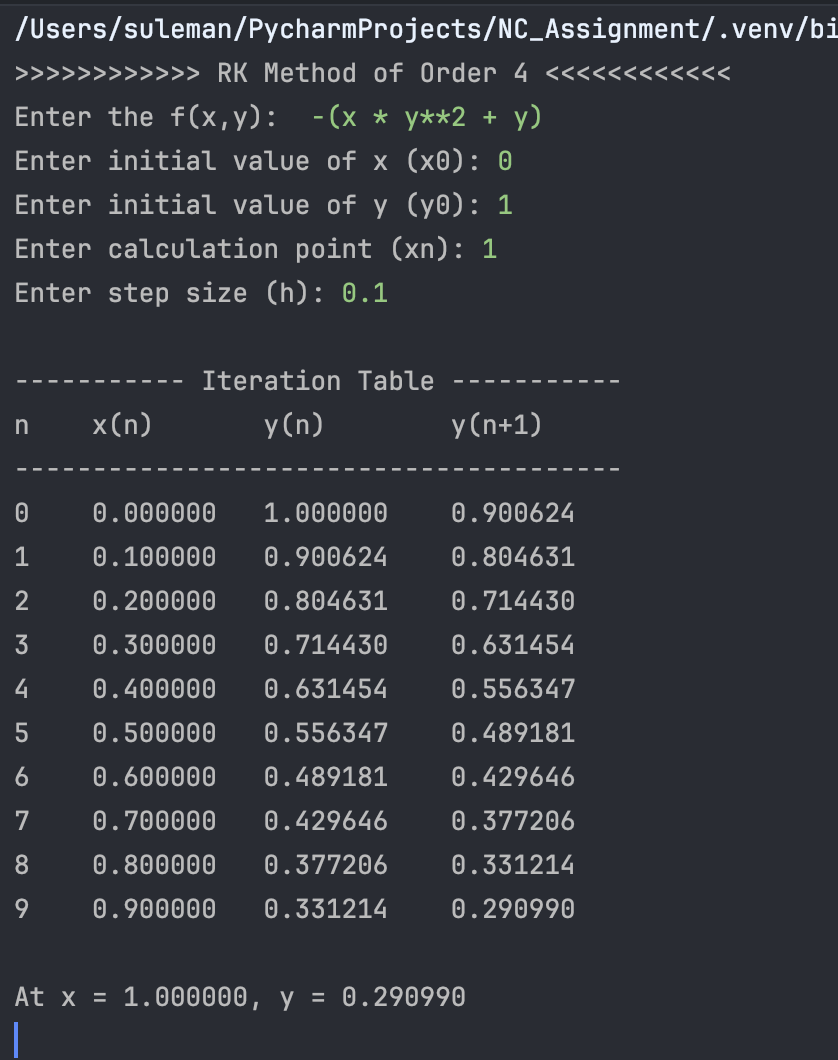
initial value of x: 0

initial value of y: 1

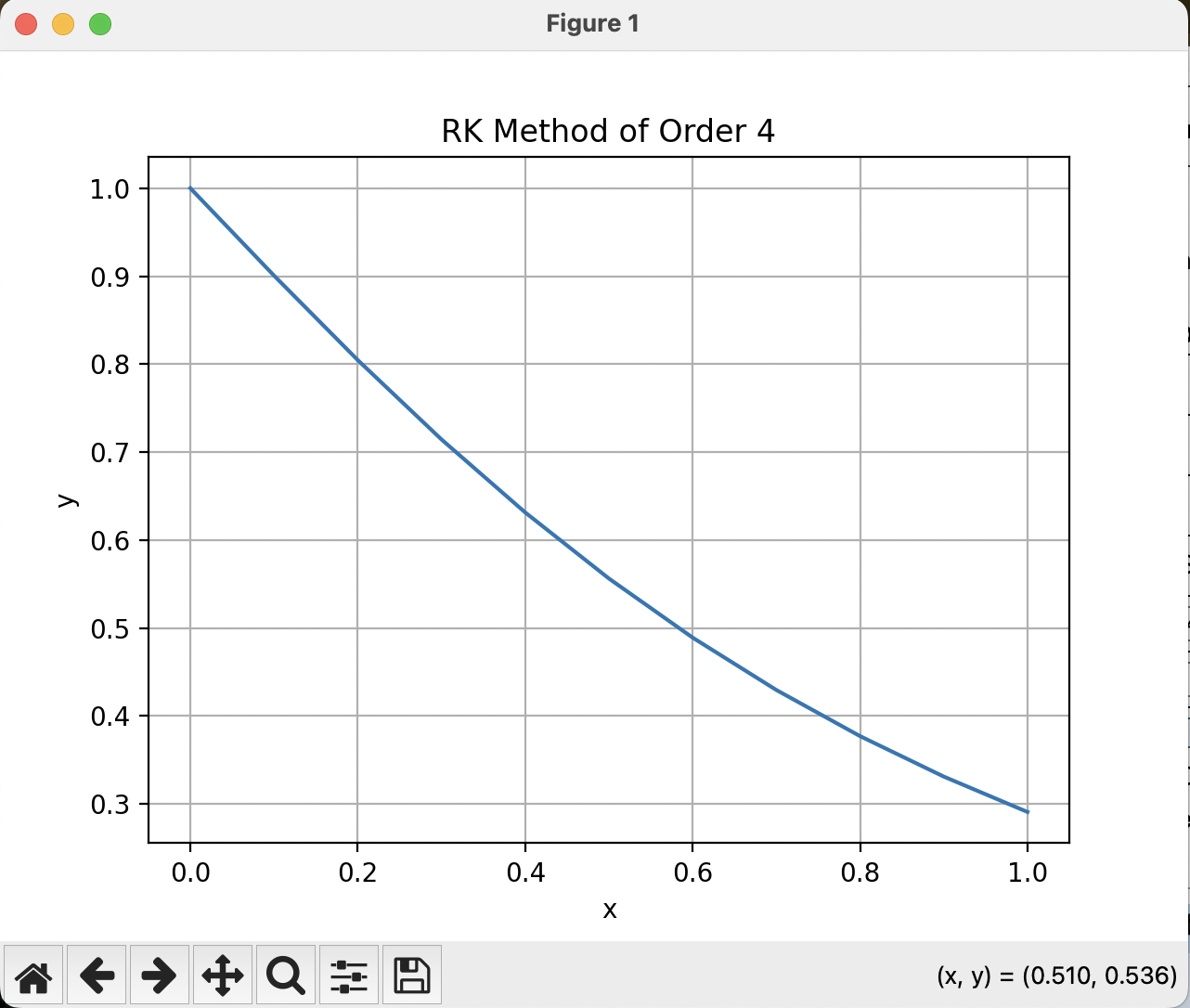
step size (h): 0.1

calculation point (xn): 1

### Output



### Graph Plot



# Question 03

## RK Method of Order 4 For 1st Order Diff-Eq

## Code

*import* matplotlib.pyplot *as* plt

*# Created empty lists to keep track of values for plotting*

x\_values = []

y\_values = []

z\_values = []

*def* f(x, y, z):

*return* eval(equation\_f) *# Evaluate the user-provided equation for f(x, y, z)*

*def* g(x, y, z):

*return* eval(equation\_g) *# Evaluate the user-provided equation for g(x, y, z)*

*def* rk4(x0, y0, z0, xn, h):

*"""*

*Runge-Kutta Fourth Order method implementation.*

*Args:*

*x0 (float): Initial value of x.*

*y0 (float): Initial value of y.*

*z0 (float): Initial value of z.*

*xn (float): Calculation point where y is evaluated.*

*h (float): Step size.*

*"""*

print("\n----------- Iteration Table -----------")

print("n\t x(n)\t\ty(n)\t\tz(n)\t\ty(n+1)\t\tz(n+1)")

print("---------------------------------------")

*# n (int): Number of steps (Iterations).*

n = int((xn - x0) / h)

x\_values.append(x0)

y\_values.append(y0)

z\_values.append(z0)

*for* i *in* range(n):

k1 = h \* f(x0, y0, z0)

l1 = h \* g(x0, y0, z0)

k2 = h \* f(x0 + h / 2, y0 + k1 / 2, z0 + l1 / 2)

l2 = h \* g(x0 + h / 2, y0 + k1 / 2, z0 + l1 / 2)

k3 = h \* f(x0 + h / 2, y0 + k2 / 2, z0 + l2 / 2)

l3 = h \* g(x0 + h / 2, y0 + k2 / 2, z0 + l2 / 2)

k4 = h \* f(x0 + h, y0 + k3, z0 + l3)

l4 = h \* g(x0 + h, y0 + k3, z0 + l3)

k = (k1 + 2 \* k2 + 2 \* k3 + k4) / 6

l = (l1 + 2 \* l2 + 2 \* l3 + l4) / 6

yn = y0 + k

zn = z0 + l

y\_values.append(yn)

z\_values.append(zn)

print(i, "\t", f"{x0:.6f}\t{y0:.6f}\t{z0:.6f}\t{yn:.6f}\t{zn:.6f}")

y0 = yn

x0 += h

z0 += l

x\_values.append(x0)

*return* x\_values, y\_values

print(">>>>>>>>>>>> RK Method of Order 4 for 1st Order Diff-Eq <<<<<<<<<<<<")

*# main()*

*# User inputs*

equation\_f = input("Enter the f(x,y,z): ")

equation\_g = input("Enter the g(x,y,z): ")

x0 = float(input("Enter initial value of x (x0): "))

y0 = float(input("Enter initial value of y (y0): "))

z0 = float(input("Enter initial value of z (z0): "))

xn = float(input("Enter calculation point (xn): "))

step\_size = float(input("Enter step size (h): "))

*# Call RK4 method*

x\_values, y\_values = rk4(x0, y0, z0, xn, step\_size)

*# Plotting*

plt.plot(x\_values, y\_values)

plt.xlabel('x')

plt.ylabel('y')

plt.title('RK Method of Order 4 for 1st Order Diff-Eq')

plt.grid(*True*)

plt.show()

*# f(x,y,z): z*

*# g(x,y,z): -4\*z -4\*y*

*# initial value of x (x0): 0*

*# initial value of y (y0): 0*

*# initial value of z (z0): 1*

*# calculation point (xn): 0.2*

*# step size (h): 0.1*

*# f(x,y,z): z*

*# g(x,y,z): 1 + x\*y - x\*\*2\*z*

*# initial value of x (x0): 0*

*# initial value of y (y0): 1*

*# initial value of z (z0): 0*

*# calculation point (xn): 0.5*

*# step size (h): 0.1*

### Sample Problem 01

f(x,y,z): z

g(x,y,z): -4\*z -4\*y

initial value of x (x0): 0

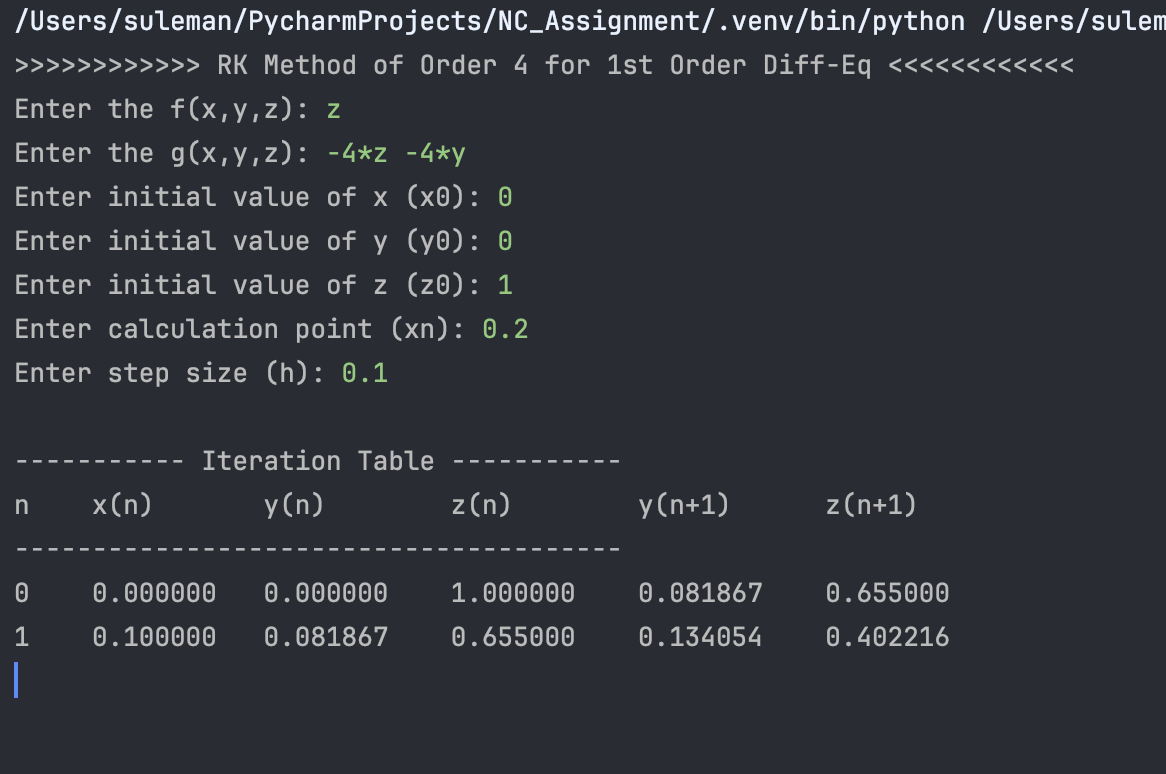
initial value of y (y0): 0

initial value of z (z0): 1

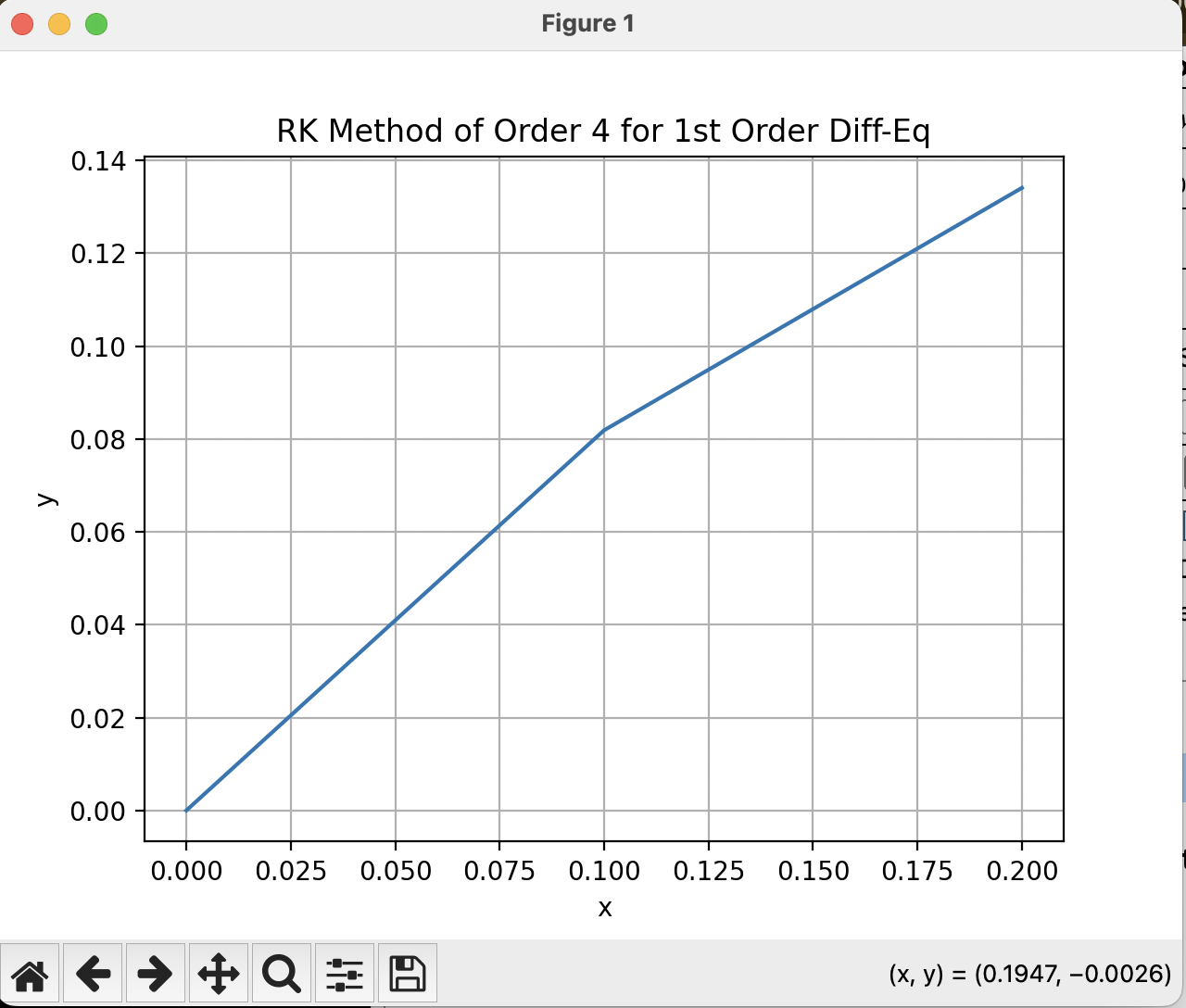
calculation point (xn): 0.2

step size (h): 0.1

### Output



### Graph Plot



### Sample Problem 02

f(x,y,z): z

g(x,y,z): 1 + x\*y - x\*\*2\*z

initial value of x (x0): 0

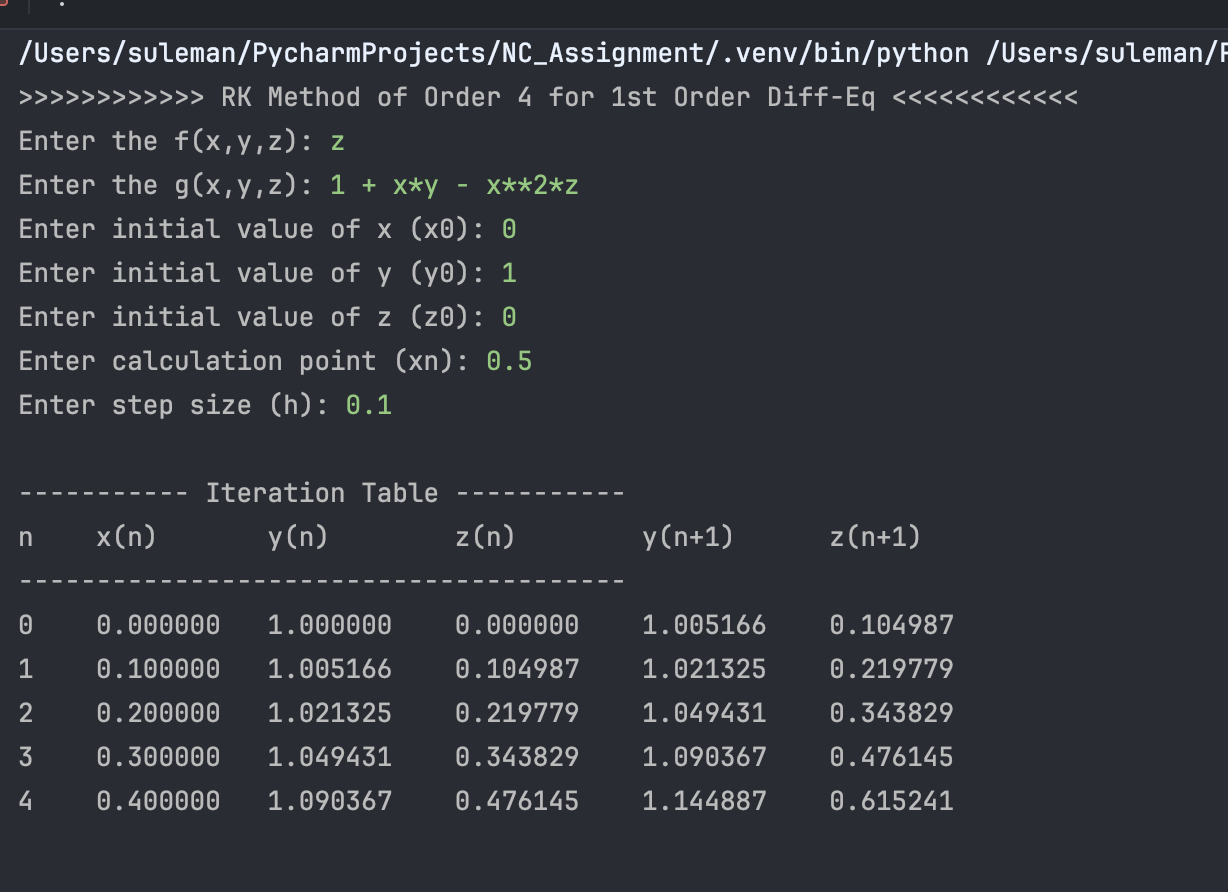
initial value of y (y0): 1

initial value of z (z0): 0

calculation point (xn): 0.5

step size (h): 0.1

### Output



### Graph Plot

