Name : Pristian Budi Dharmawan

ID : 2501983105

Study Case A

Line	Code	Description
1	import numpy as np	Importing library numpy dan diberikan nama variabel atau typedef sebagai **np**
2		
3		
4	def f(x, y):	Fungsi f dengan parameter x, y. Berfungsi untuk memasukkan entry matrix ke dalam fungsi GaussianElimination. Dimana hasil return akan disimpan pada array dengan variabel temp. Selain itu nilai yang direturn berupa nilai x dan y yan dikurangi dengan formula -x*x-y*y yang mana formula ini berfungsi untuk mendapatkan nilai z dari A matrix sehingga kita bisa mendapatkan bentuk matrix yang diinginkan
5	return [x, y, 1], -x*x-y*y	
6		
7		
8	def partialPivot(A, n, k):	Dengan menggunakan metode partial pivoting. Metode ini akan menukarkan bagian "main diagonal pivot" dengan baris kedua atau bawahnya. Hal ini bisa dilihat pada ilustrasi berikut: Dari Bentuk $\left\|\begin{matrix}0 & 5 & -12\\ -9 & 0 & 4\\ 7 & 0 & 3\end{matrix}\right\|\begin{matrix}3\\3\\6\end{matrix}$ Menjadi $\left\|\begin{matrix}-9 & 0 & 4\\ 0 & 5 & -12\\ 7 & 0 & 3\end{matrix}\right\|\begin{matrix}3\\3\\6\end{matrix}$
9	absMaxIdx = k	
10	for i in range(k, n):	
11	if abs(A[i][k]) > abs(A[absMaxIdx][k]):	
12	absMaxIdx = i	
13	if absMaxIdx != k:	
14	A[[k, absMaxIdx], :] = A[[absMaxIdx, k], :]	
15		
16		

17	def printMatrix(A, n):	Fungsi ini berfungsi untuk melakukan print pada Matrix A dan Matrix B. Sehingga kita bisa melihat perubahan pada setiap step yang dilakukan dalam Gaussian Elimination, yaitu membuat segitiga bawah menjadi 0
18	print("A matrix:")	
19	for i in range(n):	
20	for j in range(n):	
21	print(f"{A[i][j]:.4f}", end=" ")	
22	print("")	
23	print("")	
24	print("B matrix:")	
25	for i in range(n):	
26	print(f"{A[i][n]:.4f}")	
27	# print(round(A[i][n], 4))	
28	print("")	
29		
30		
31	def GaussianElimination(A, B, pivot=True, showall=True):	Pada bagian ini, program akan melakukan partial pivoting terlebih dahulu dan dilanjutkan dengan perubahan segitiga bawah menjadi 0. Setelah melakukan Step 1 untuk merubah segitiga bawah menjadi 0, kemudian Gaussian Elimination akan dilanjutkan dengan cara subtitusi balik sehingga kita bisa mendapatkan nilai final yaitu solusi dari matrix yang diberikan.
32	n = len(A)	
33	x = np.zeros(n)	
34	A = np.append(A, B, axis=1)	
35	# Forward elimination	
36	for k in range(0, n-1): # goes down -> (n-1) steps	
37	if pivot:	
38	partialPivot(A, n, k)	
39	# A[absMaxIdx][j] == 0	
40	if A[k][k] == 0:	
41	return [False, x]	
42	if showall:	
43	print('Step {:d}:'.format(k+1))	
44	for i in range(k+1, n): # goes right -> (n-k) sub steps	
45	factor = A[i][k]/A[k][k]	
46	# Changes the whole i-th row	
47	for j in range(n+1):	
48	A[i][j] = A[i][j] - factor*A[k][j]	
49	if showall:	
50	print('Sub Step {:d}:'.format(i-k))	
51	printMatrix(A, n)	
52	# Back substitution	
53	for i in range(n-1, -1, -1):	
54	x[i] = A[i][n]	
55	for j in range(i+1, n):	
56	x[i] = x[i]-x[j]*A[i][j]	
57	if A[i][i] == 0:	
58	if(x[i] == 0):	
59	print("No unique solution.Infinite solutions.")	
60	else:	
61	print("No solution")	
62	return [False, x]	

63	x[i] = x[i]/A[i][i]	
64	return [True, x]	
65		
66		
67	def main():	Fungsi ini berfungsi sebagai fungsi utama. Fungsi ini diawali dengan alokasi nilai dari System of Linear Equation ke dalam pemanggilan fungsi f() Dilanjutkan dengan pengecekan status dari nilai yang dikembalikan pada fungsi GaussianElimination, jika status bernilai TRUE maka akan melakukan printing hasil dari nilai final yang telah dihitung pada fungsi GaussianElimination. Namun jika bersifat False maka akan melakukan print bahwa tidak ada solusi dari matrix yang diberikan
68	n = 3	
69	# n = int(input())	
70	A = np.zeros((n, n))	
71	B = np.zeros((n, 1))	
72	# for i in range(n):	
73	# A[i] = np.array([list(map(float, input().split()))], float)	
74	# for i in range(n):	
75	# B[i][0] = float(input())	
76		
77	temp = f(-2, 0)	
78	A[0], B[0] = temp[0], temp[1]	
79	temp = f(-1, 7)	
80	A[1], B[1] = temp[0], temp[1]	
81	temp = f(5, -1)	
82	A[2], B[2] = temp[0], temp[1]	
83		
84	status, x = GaussianElimination(A, B)	
85	if status:	
86	print("Solution:")	
87	for i in x:	
88	print(f"{i:.4f}")	
89	else:	
90	print("No solution exists")	
91		
92		
93	if __name__ == '__main__':	Sebagai typedef dari bentuk main
94	main()	
95		
96	'''	Bagian ini merupakan sebuah string penjelasan System of Linear Equations yang akan diberlakukan Gaussian Elimination
97	-2a + 0b + c = -4	
98	-a + 7b + c = -50	
99	5a - b + c = -26	
100	3	
101	-2 0 1	
102	-1 7 1	
103	5 -1 1	
104	-4	
105	-50	
106	-26	
107	'''	

Study Case B

Line	Code	Description
1	import numpy as np	Importing library dengan nama variabel np, plt, dan pd
2	import matplotlib.pyplot as plt	
3	import pandas as pd	
4		
5		
6	def linear(x, y):	Fungsi ini akan menghitung nilai regresi dari setiap element yang ada pada dalam array Xi dan Yi. Dengan beberapa fungsi dari library panda, kita bisa mendapatkan nilai regresi tersebut. Seperti penambahan element dari np.append; pembuatan tabel computer dengan fungsi pd.DataFrame untuk menghasilkan array 2 dimensi; dan df.index untuk mengakses data pada bagian index tertentu di dalam pd.DataFrame
7	n = np.size(x)	
8	p = np.sum(x*x)	
9	q = np.sum(x)	
10	r = np.sum(x*y)	
11	s = np.sum(y)	
12	a_1 = (n*r-q*s) / (n*p - q*q)	
13	a_0 = (p*s-q*r) / (n*p - q*q)	
14		
15	data = {'x': np.append(x, q), 'y': np.append(	
16	y, s), 'x^2': np.append(x*x, p), 'xy': np.append(x*y, r)}	
17	df = pd.DataFrame(data)	
18	df.index = np.arange(1, len(df)+1)	
19	print(df)	
20	return (a_0, a_1)	
21		
22		
23	def drawGraph(x, y, y_pred):	Fungsi ini berfungsi untuk membuat sebuah graphs pada program kita setelah dijalankan. Dapat dilihat scatter berfungsi untuk melihat seberapa jauh nilai x dan y yang ingin ditampilkan dan pol sebagai penanda atau legend dalam graph yang telah dibuat
24	plt.scatter(x, y)	
25	plt.plot(x, y_pred)	
26	plt.show()	
27		
28		
29	def main():	Dengan membuat sebuah baris nilai Xi dan Yi dan menempatkan pada array, kita dapat memanggil fungsi linear(x, y) untuk menghitung nilai regresi dari nilai-nilai Xi dan Yi
30	x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])	
31	y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])	
32	a_0, a_1 = linear(x, y)	
33	drawGraph(x, y, a_0 + a_1*x)	
34		
35		
36	if __name__ == "__main__":	Sebagai typedef dari bentuk main

37	main()	

Study Case C

Line	Code	Description
DISCLAIMER: Source code GitHub untuk Newton Divided Difference memiliki sebuah error "IndentationError: unexpected independent" Sehingga saya melakukan format ulang document (Shift+Alt+F) di Visual Studio Code yang mana error ini terdapat pada line 2-5.		
1	def proterm(i, value, x):	Fungsi ini adalah kelanjutan dari fungsi applyFormula() yaitu untuk mengimplementasikan rumus ke dalam program
2	pro = 1	
3	for j in range(i):	
4	pro = pro * (value - x[j])	
5	return pro	
6		
7		
8	def dividedDiffTable(x, y, n):	
9		
10	for i in range(1, n):	
11	for j in range(n - i):	
12	y[j][i] = ((y[j][i - 1] - y[j + 1][i - 1]) /	
13	(x[j] - x[i + j]))	
14	return y	
15		
16		
17	def applyFormula(value, x, y, n):	Fungsi ini adalah fungsi untuk membuat formula menghitung Newton Divided dalam bentuk program.
18		
19	sum = y[0][0]	
20		
21	for i in range(1, n):	
22	sum = sum + (proterm(i, value, x) * y[0][i])	
23		
24	return sum	
25		
26		
27	def printDiffTable(y, n):	Fungsi ini berfungsi untuk menghasilkan bentuk dari Newton Divided Difference Table "Inverted 90° Triangle"
28		
29	for i in range(n):	
30	for j in range(n - i):	
31	print(round(y[i][j], 4), "\t",	
32	end = " ")	
33		
34	print("")	
35		
36		
37	n = 5; # manual input	Setelah fungsi-fungsi diatas, program ini menggunakan global
38	y = [[0 for i in range(10)]	

39	for j in range(10)]	syntax yang mana tidak ada fungsi utama. Sehingga biasa saya sebut “Self-Service” karena tanpa adanya fungsi utama, program ini dapat berjalan dengan baik dengan bypass nilai-nilai atau nama variabel yang sama pada variabel global ke dalam setiap fungsi yang ada. Jadi kesimpulan dari line 37-58 adalah “Main Function” dari program ini
40	x = [-1, 0, 3, 6, 7]; # manual input	
41		
42	y[0][0] = 3; # manual input	
43	y[1][0] = -6	
44	y[2][0] = 39	
45	y[3][0] = 822	
46	y[4][0] = 1611	
47		
48		
49	y = dividedDiffTable(x, y, n);	
50		
51	printDiffTable(y, n)	
52		
53	# value to be interpolated	
54	value = 2	
55		
56	# printing the value	
57	print("\nValue at", value, "is",	
58	round(applyFormula(value, x, y, n), 2))	