**Aplikasi Penyusunan Rencana Kuliah dengan Topological Sort**

**(Penerapan Decrease and Conquer)**

**“PeRaU”**

**Laporan Tugas Kecil 2 IF2211 Strategi Algoritma**

**Disusun sebagai syarat tugas kecil 2 mata kuliah IF2211 Strategi Algoritma**

**Semester II tahun 2020/2021**

**Disusun oleh:**

**Ferdy Irawan Firdaus (13519030)**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO & INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2021**

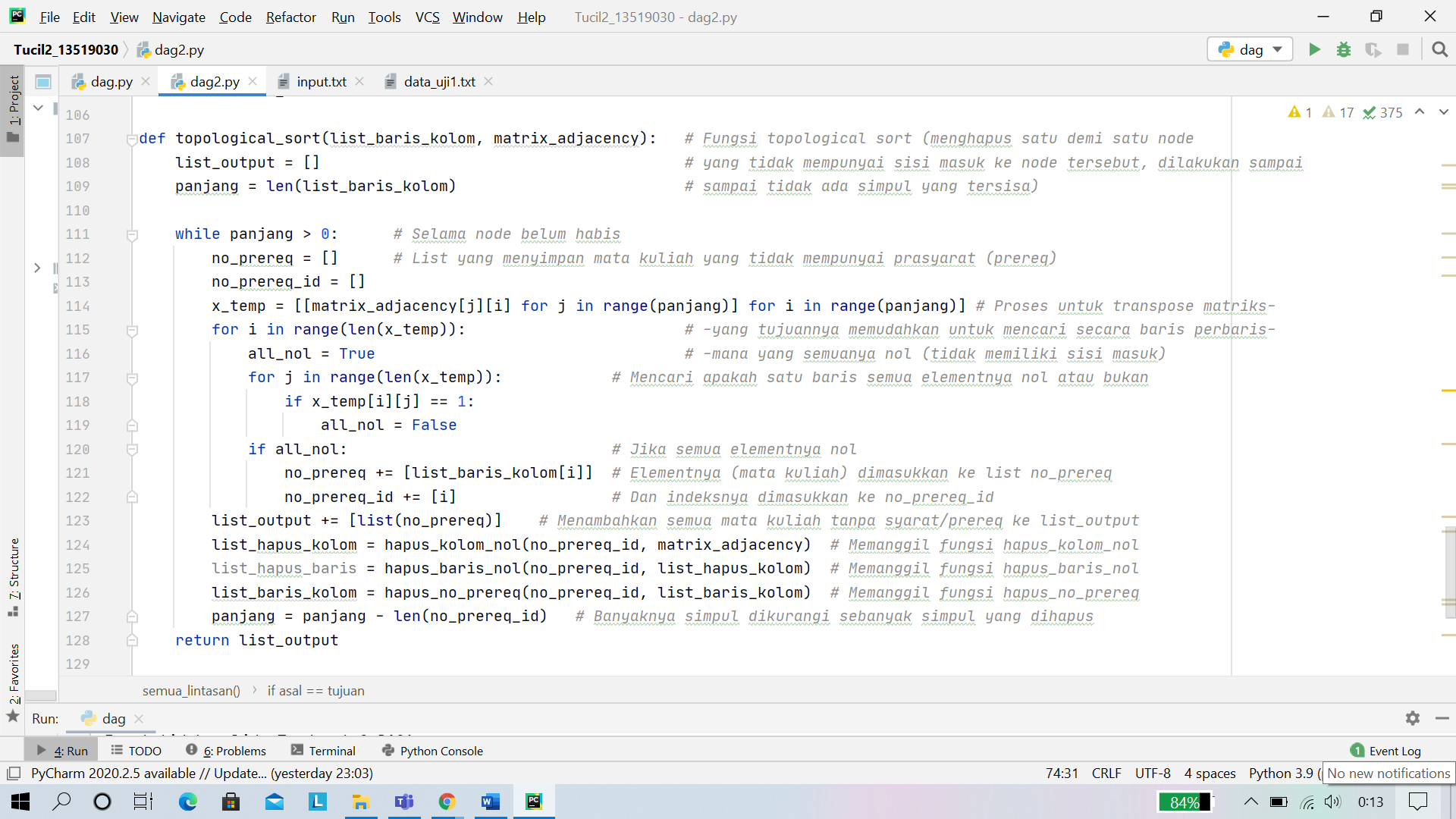
**BAB I**

**ALGORITMA TOPOLOGICAL SORT**

Ada beberapa fungsi maupun prosedur yang berkaitan dengan algortima *topological sort* dan memanfaatkan pendekatan *decrease and conquer.*

* 1. **Fungsi topological\_sort()**

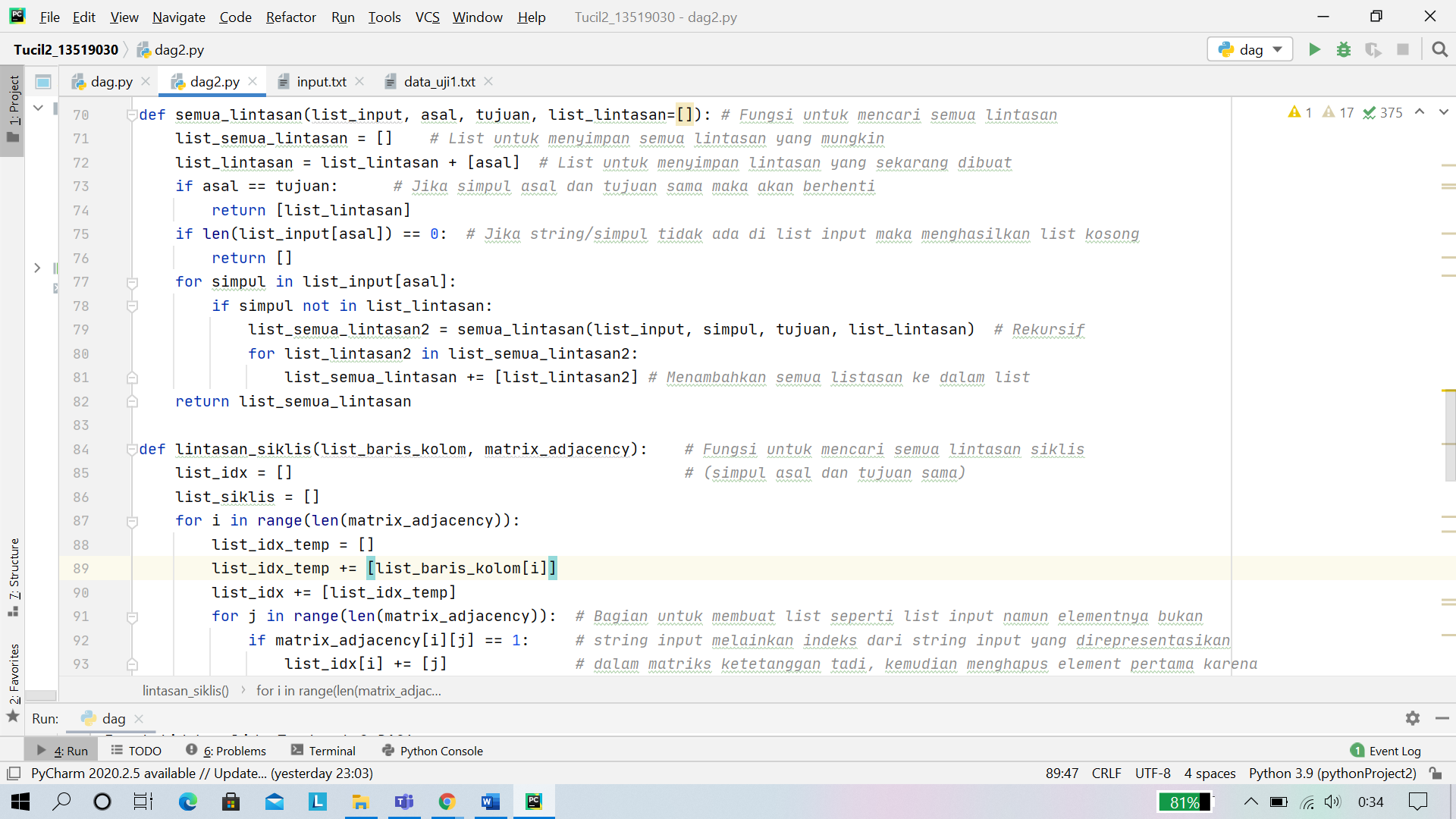
Dalam fungsi topological\_sort() terdapat dua parameter yaitu list\_baris\_kolom (list yang berisi semua mata kuliah secara unik atau tidak ada yang dobel) dan matrix\_adjacency (matriks dengan baris dan kolomnya adalah semua mata kuliah atau node/simpul yang barisnya merupakan simpul asal dengan sisi keluar dari simpul sedangkan kolom adalah simpul tujuan dengan sisi masuk ke simpul tersebut dan elemennya merupakan representasi dari sisi berarah yang berupa angka 1 jika kedua simpul dihubungkan oleh sebuah sisi dan 0 jika kedua simpul tidak terhubung oleh sebuah sisi). Fungsi ini bertujuan untuk melakukan topological sort dengan kata lain mengurangi/menghapus node yang tidak mempunyai sisi masuk atau derajat sisi masuk adalah nol sampai semua matkul terpilih dalam delapan semester. Representasi dalam matriks ketetanggaan adalah dapat dilihat di semua kolomnya, jika dalam satu simpul semua kolomnya bernilai 0 maka simpul tersebut tidak mempunyai satupun sisi masuk namun sebaliknya jika satu saja ada elemen dalam kolom tersebut bernilai 1 maka terdapat sisi masuk artinya mata kuliah tersebut memiliki prasyarat mata kuliah lain. Kemudian setelah mencari kolom mana yang semua elemennya bernilai 0 maka dilakukan penghapusan kolom pada indeks tersebut lalu dilanjutkan penghapusan baris pada indeks yang sama dengan indeks kolom. Langkah ini dilakukan secara terus menerus sampai tidak ada mata kuliah yang tersisa. Dari uraian tersebut maka algoritma dalam fungsi topological\_sort sudah menerapkan pendekatan algoritma decrease and conquer yaitu mereduksi secara bertahap dari matriks kemudian dicari mana yang tidak mempunyai sisi masuk dan menghapusnya dan pengurangan elemen ini dilakukan secara berulang sampai mata kuliah tidak bersisa.



**Gambar 1.1 Fungsi topological\_sort()**

* 1. **Fungsi semua\_lintasan()**

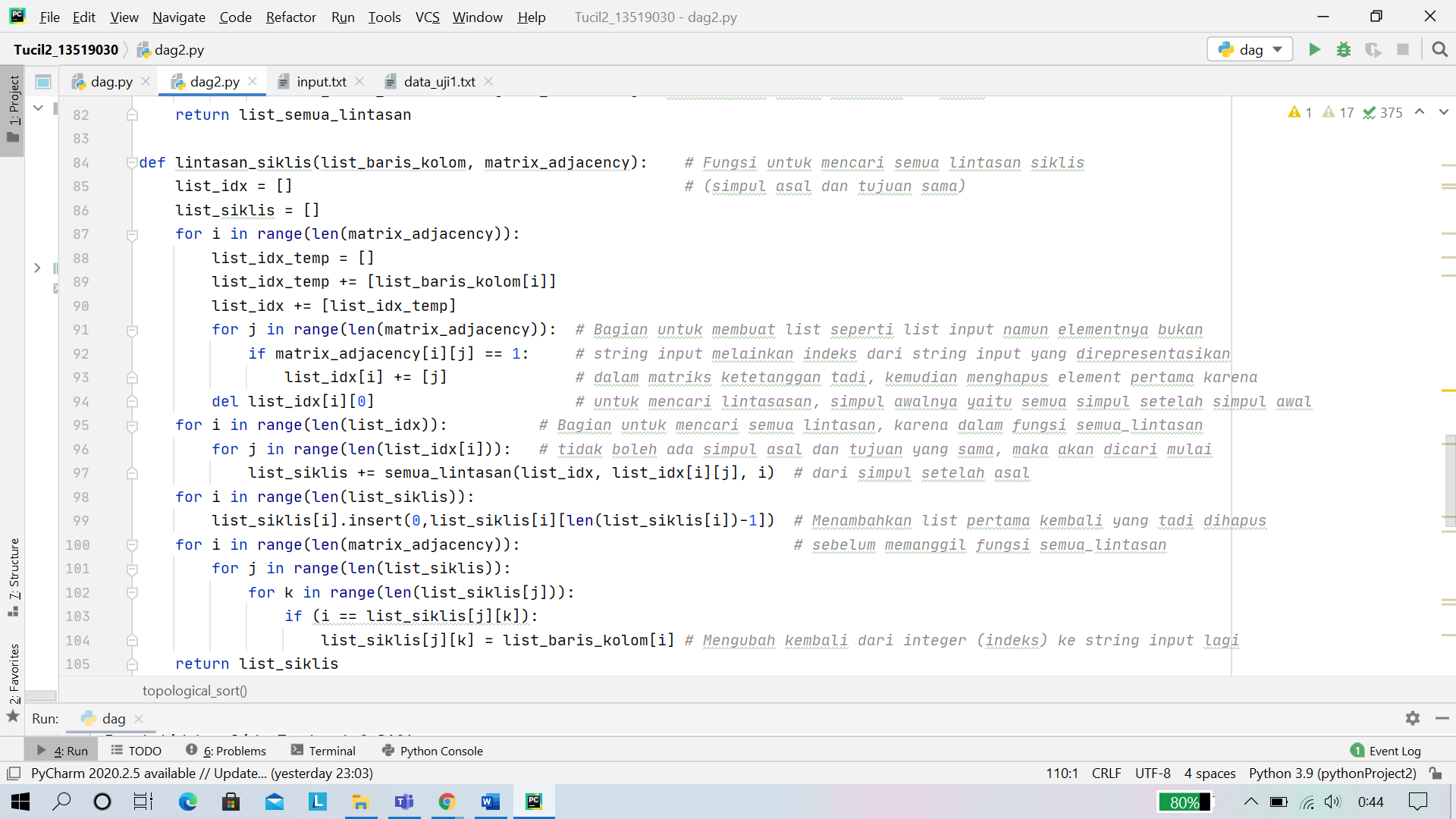
Dalam fungsi semua\_lintasan() ini bertujuan mencari semua lintasan dengan memasukkan list input, simpul asal, dan simpul tujuan. Sehingga nantinya fungsi ini akan digunakan dalam fungsi lintasan\_siklis() yang membantu untuk mengetahui apakah input merupakan graf siklis/sekuler atau DAG dan jika graf tidak DAG maka akan menyimapn semua lintasan yang bermula dan berakhir pada simpul yang sama, namun dalam fungsi semua\_lintasan ini terdapat kondisi simpul awal dan tujuan tidak boleh sama namun hal ini akan diatasi dalam fungsi lintasan\_siklis. Fungsi ini menggunakan metode rekursif.



**Gambar 1.2 Fungsi semua\_lintasan()**

* 1. **Fungsi lintasan\_siklis()**

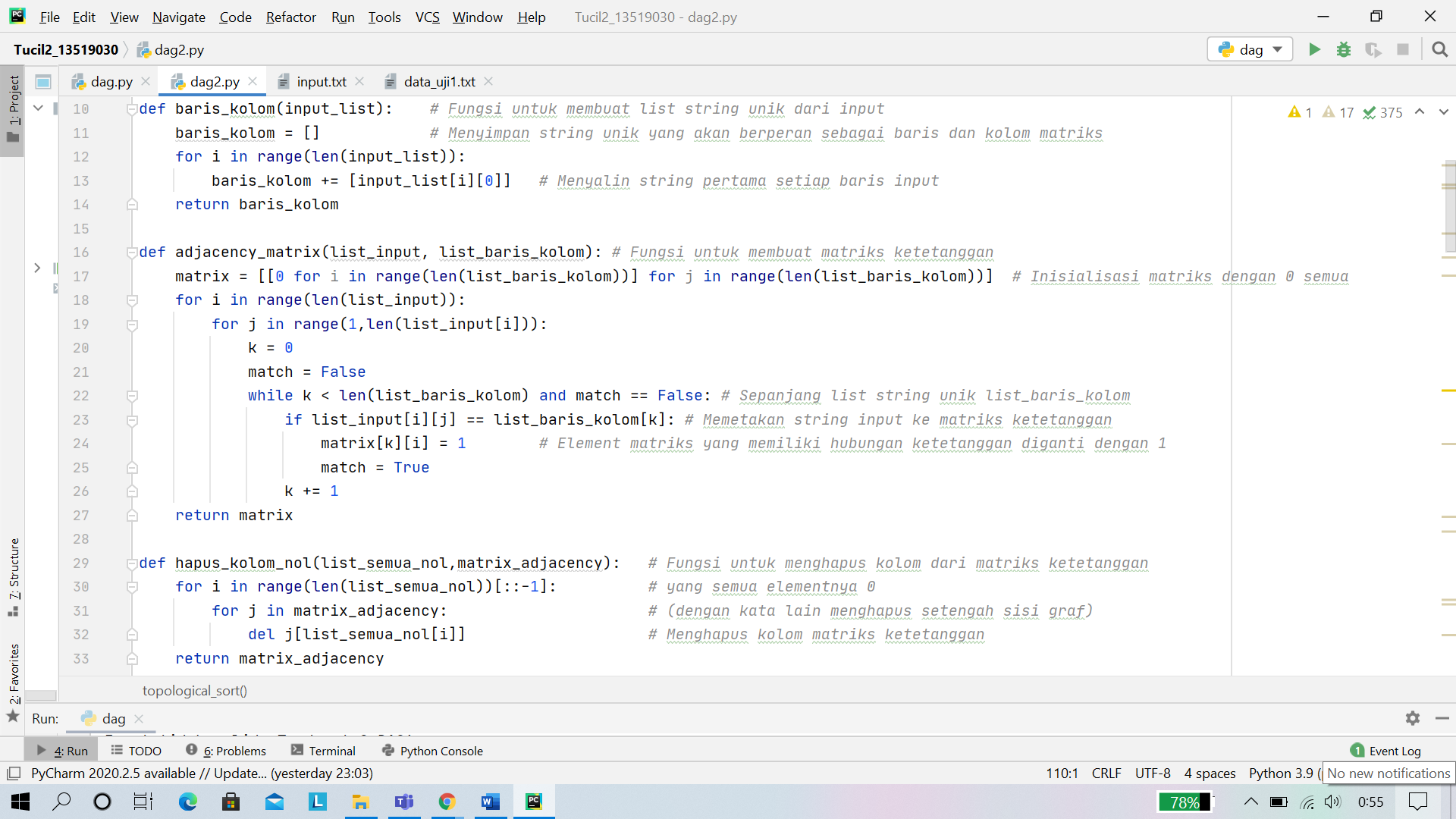
Fungsi ini bertujuan mencari semua lintasan sekuler/siklis jika input dari pengguna bukan DAG dan nantinya akan ada kondisi dalam main program jika input bukan DAG maka akan menampilkan pesan bahwa input bukan DAG dan ditampilkan juga semua lintasan sekuler.siklis yang mungkin ada dalam graf input. Seperti yang dijelaskan pada fungsi semua\_lintasan, fungsi lintasan\_siklis ini memanfaatkan fungsi semua\_lintasan dengan melakukan pengecekan mencari semua mata kuliah sebagai simpul awal sekaligus simpul tujuannya, namun pada fungsi semua\_lintasan tidak boleh jika simpul awal dan akhirnya sama maka dilakukan pencarian mulai dari satu tingkat di bawah simpul awal yang sebenarnya. Misalnya dari input terdapat lima mata kuliah yaitu a, b, c, d, dan e maka dilakukan iterasi pengecekan apakah mungkin ada lintasan sekuler dimulai dari simpul awalnya a dan simpul tujuannya a juga, lalu simpul awalnya b dan simpul tujuannya b juga, hal ini dilakukan sampai simpul terakhir yaitu e, namun karena ada kondisi dalam fungsi semua\_lintasan bahwa simpul awal tidak sama dengan simpul tujuan maka jika dalam simpul a terdapat beberapa sisi yang keluar contohnya ada tiga sisi keluar yaitu keluar ke simpul c, d, dan e maka pengecekan dimulai dari ketiga simpul tersebut.



**Gambar 1.3 Fungsi lintasan\_siklis()**

* 1. **Fungsi adjacency\_matrix()**

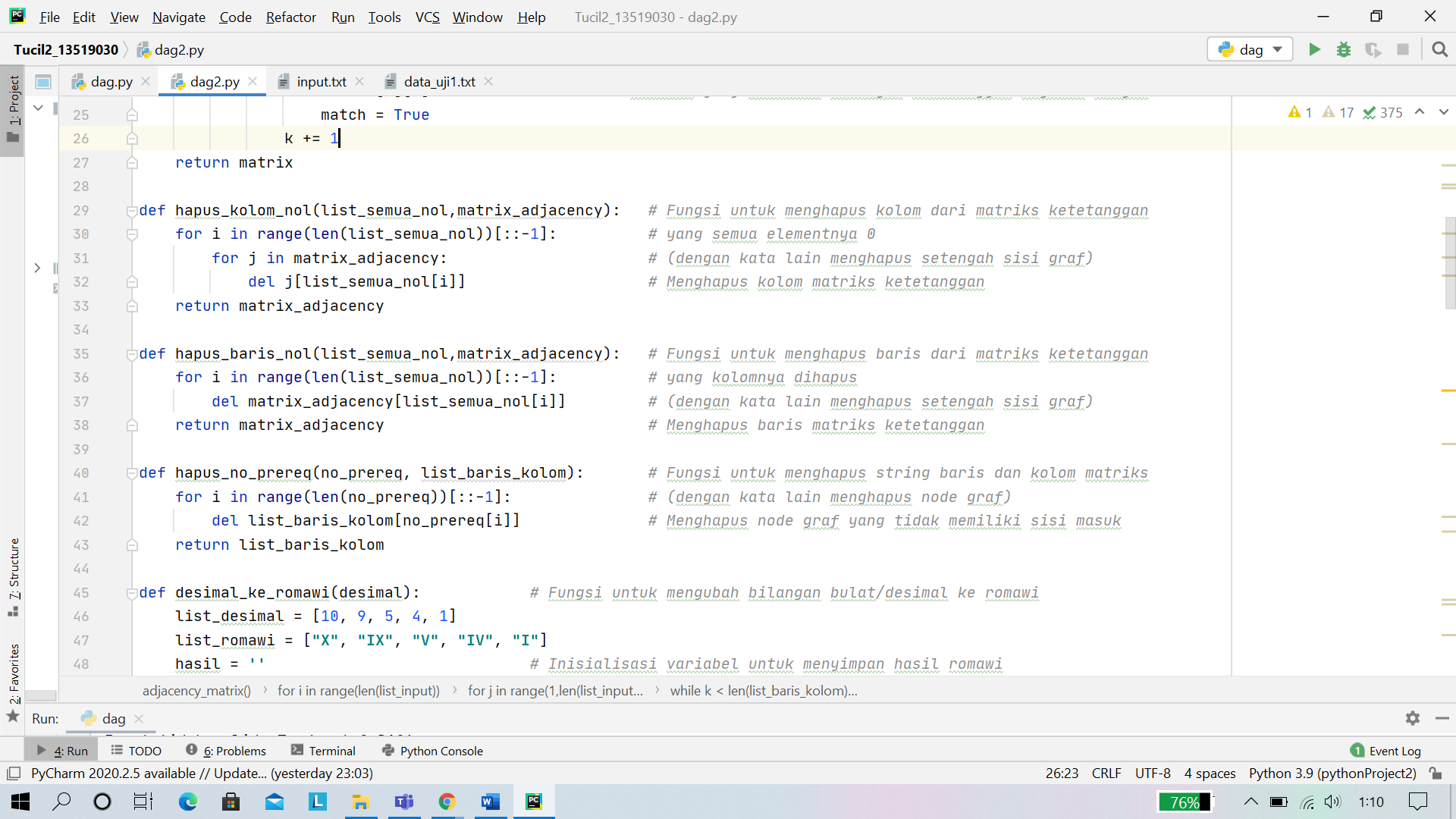
Dalam fungsi ini bertujuan untuk membuat matriks ketetanggan dengan menerima dua list yaitu list\_input (list hasil copy dari file.txt ke dalam list yang disalin baris perbaris) dan list\_baris\_kolom yaitu berisi string mata kuliah (unik atau tidak ada yang dobel) yang akan menjadi baris dan kolom atau sebagai simpul graf matriks ketetanggaan, sedangkan elemennya merepresentasikan sisi berarah graf berisi angka 0 jika kedua simpul tidak bertetanggan dan 1 jika kedua simpul bertetangga, arah sisi ini dimulai dari simpul yang berada di baris dan menuju ke simpul di kolom misalnya pada indeks (2,1) elemennya bernilai 1 maka sisi keluar dari simpul dengan indeks kedua (dari contoh sebelumnya indeks 0 mata kuliah a, indeks 1 mata kuliah b, indeks 2 mata kuliah c dan seterusnya) dan sisi masuk ke simpul dengan indeks 1 artinya simpul c dan b bertetanggan yang dihubungkan simpul dengan arah dari simpul c dan masuk ke simpul b.



**Gambar 1.4 Fungsi adjacency\_matrix()**

* 1. **Fungsi hapus\_kolom\_nol(), hapus\_baris\_nol(), dan hapus\_no\_prereq()**

Ketiga fungsi ini diimplementasikan dalam fungsi topological\_sort() yang mana fungsi hapus\_kolom\_nol() berfungsi untuk menghapus kolom yang semua elemennya bernilai nol artinya mata kuliah dalam indeks kolom ini tidak mempunyai prasyarat. Fungsi hapus\_baris\_nol() berfungsi untuk menghapus kolom yang indeksnya sama dengan indeks kolom yang tidak memiliki prasyarat tadi. Dan terakhir adalah fungsi hapus\_no\_prereq() yang berfungsi menghapus mata kuliah yang tidak memiliki prasyarat tadi dari list mata kuliah yang unik tadi.



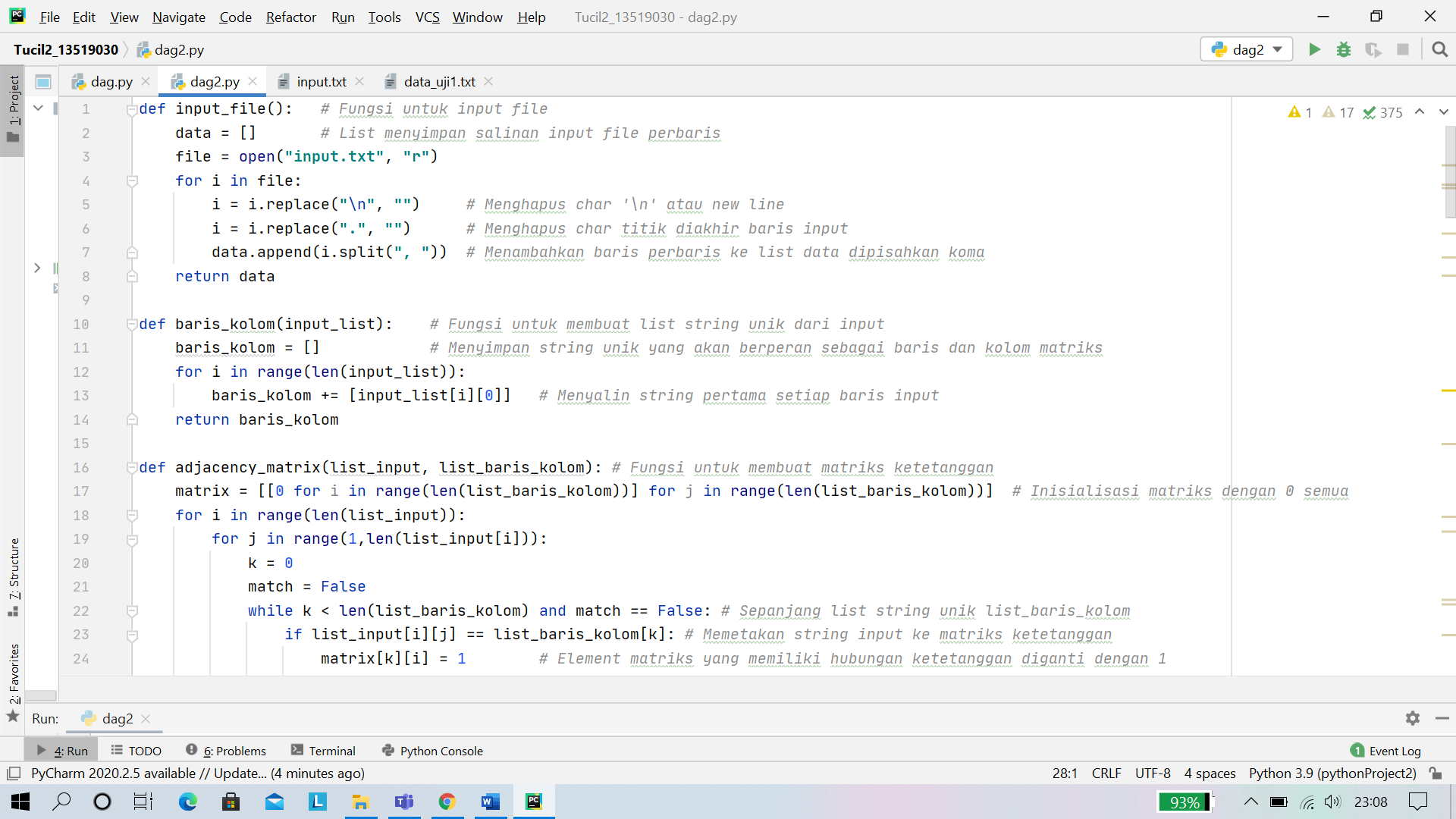
**Gambar 1.5 Fungsi hapus\_kolom\_nol(), hapus\_baris\_nol(), dan hapus\_no\_prereq()**

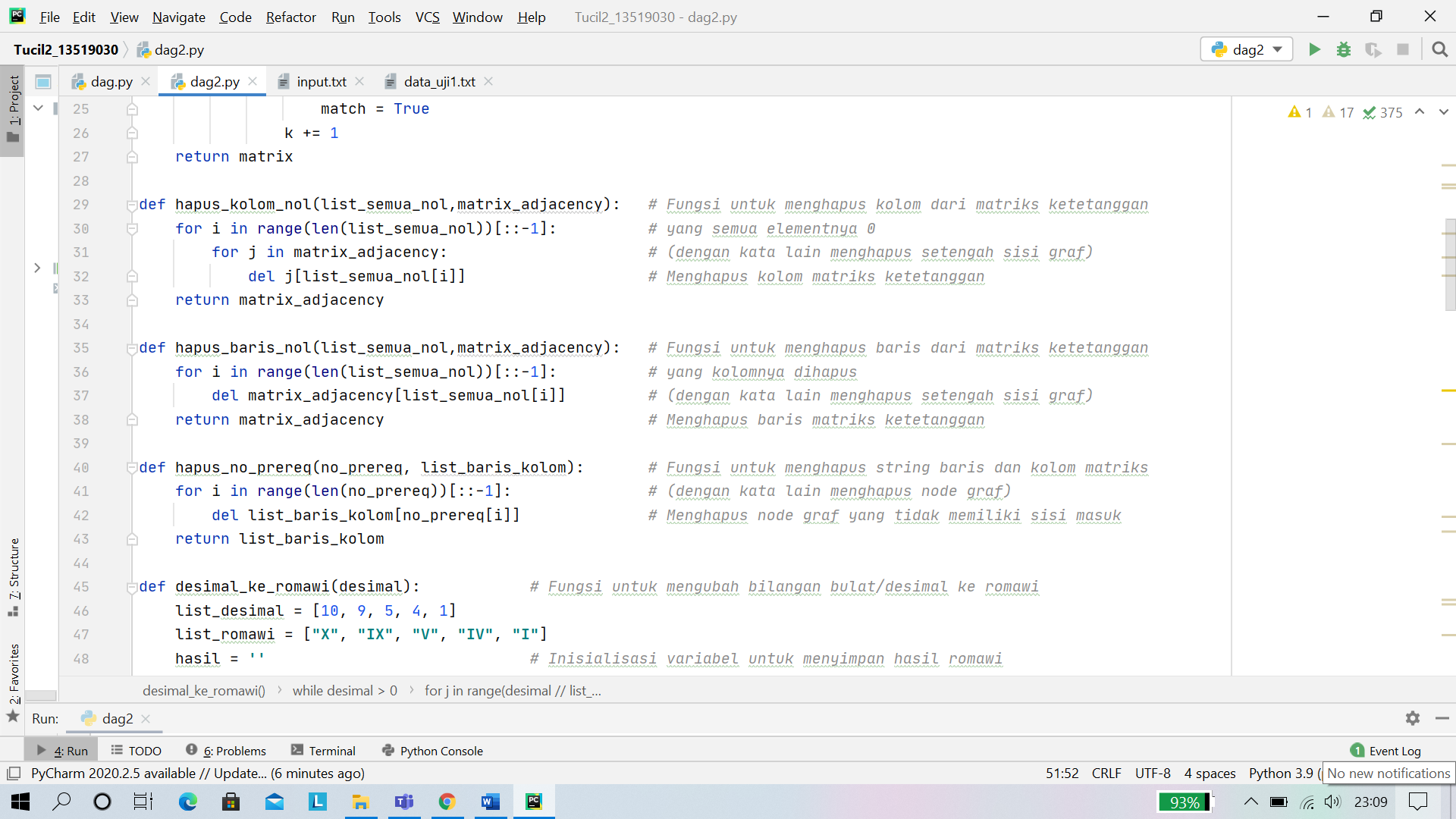
**BAB II**

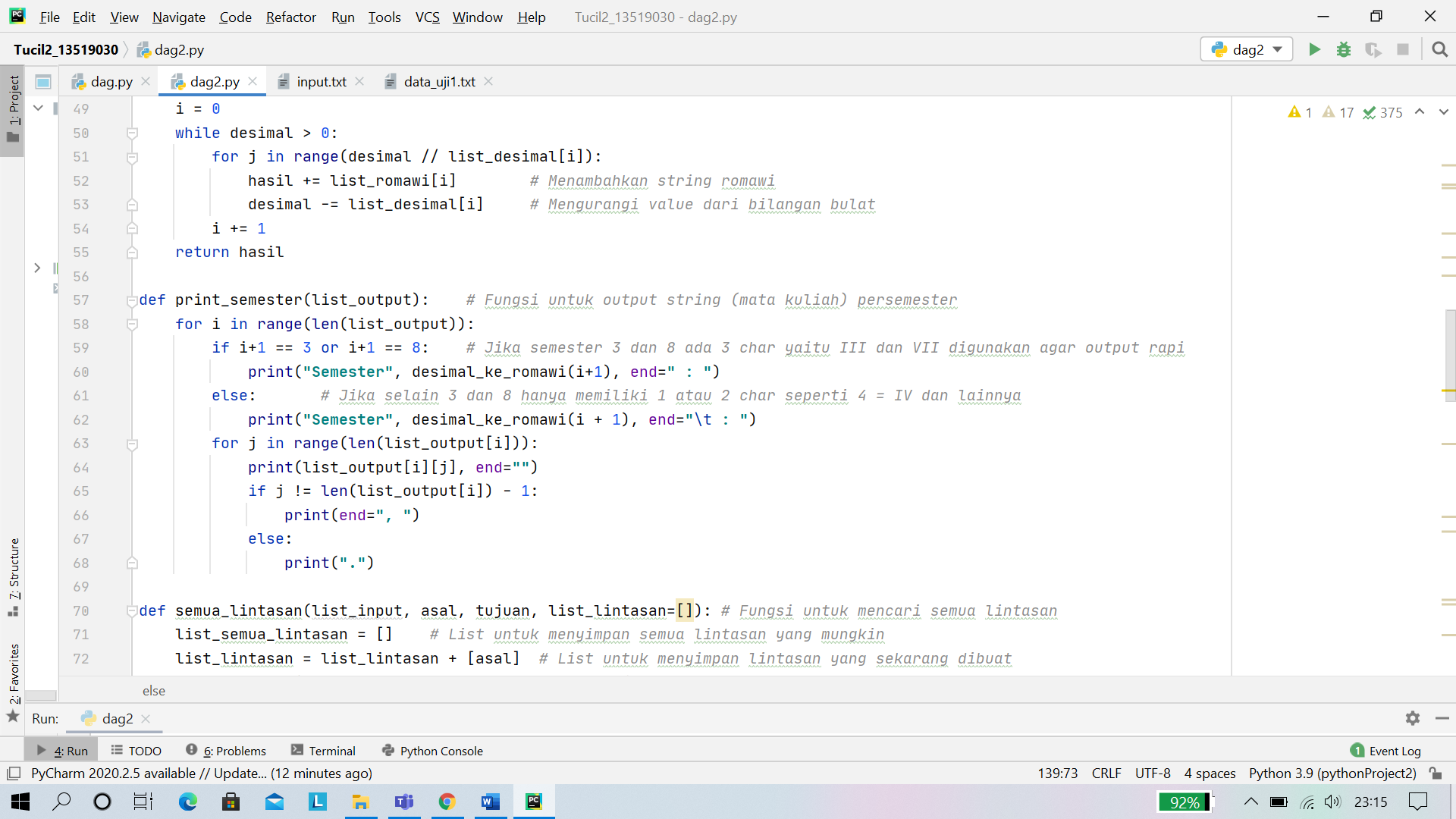
**SOURCE CODE PROGRAM**

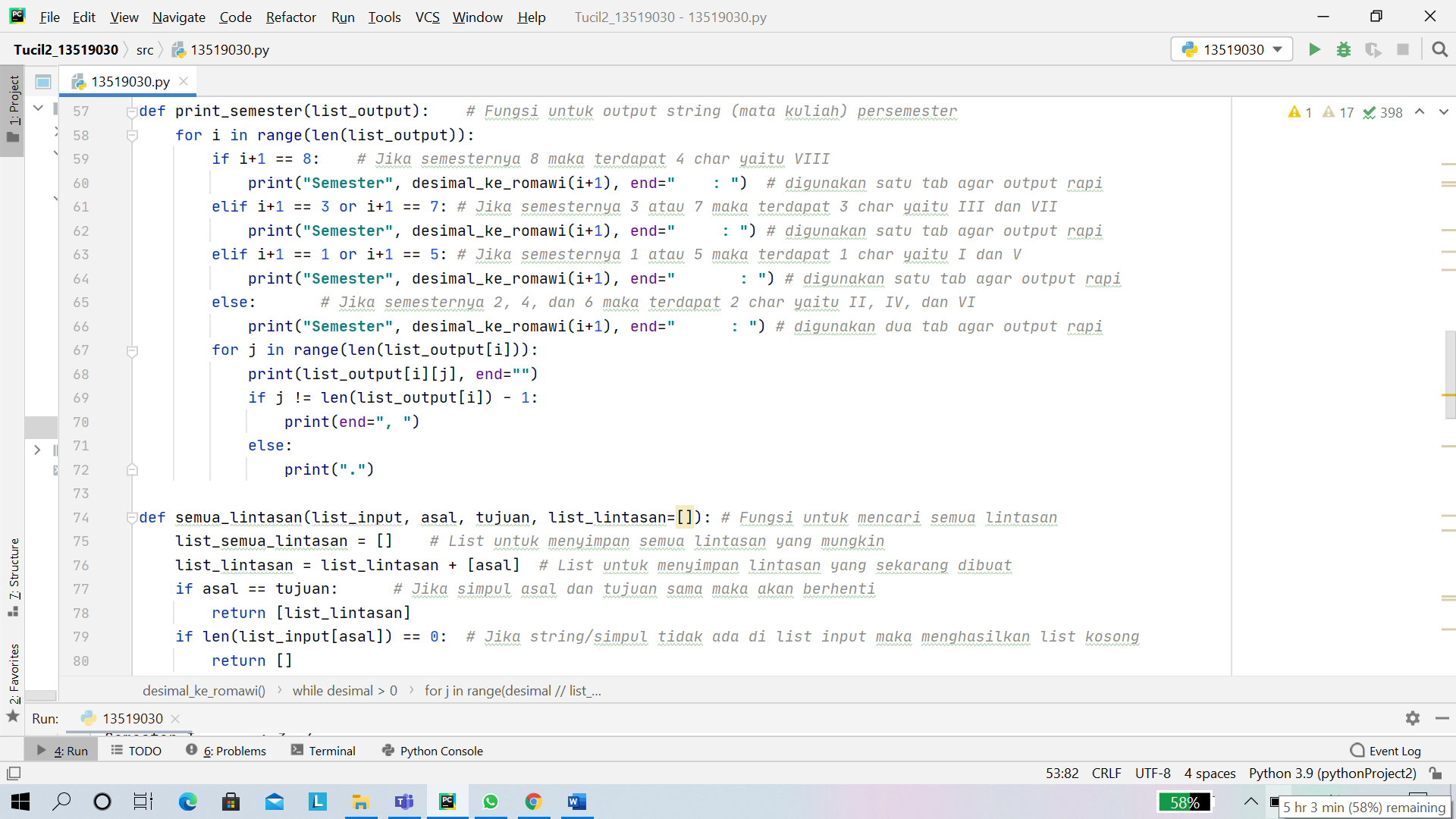
Ada dua bagian yaitu tangkap layar (dengan komentar) dan hasil *copy paste source code* (tanpa komentar) karena jika *copy paste* dengan komentar maka hasilnya tidak bagus/rapi.

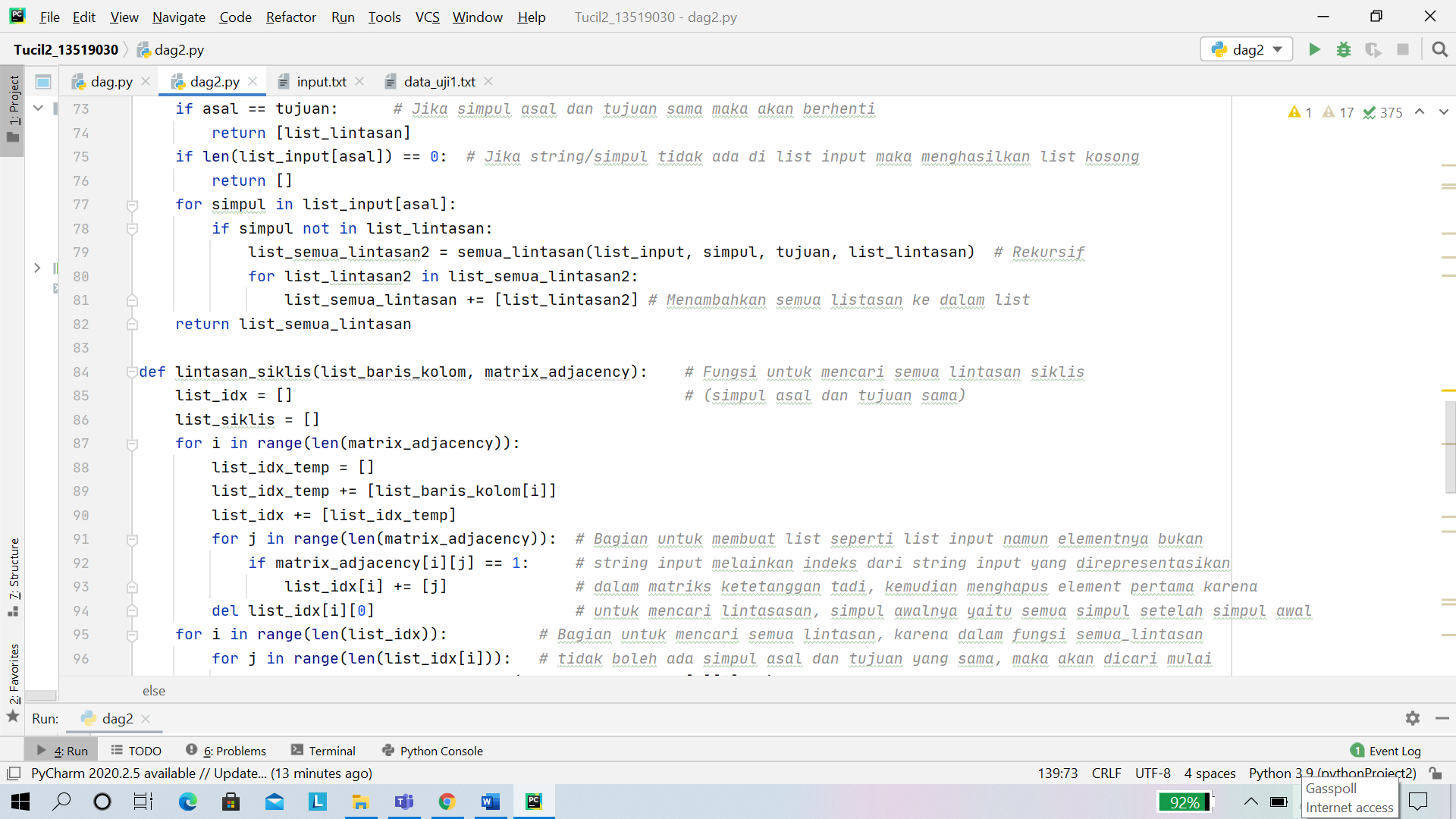
**2.1. Tangkap Layar (*Screenshot*) *Source Code***

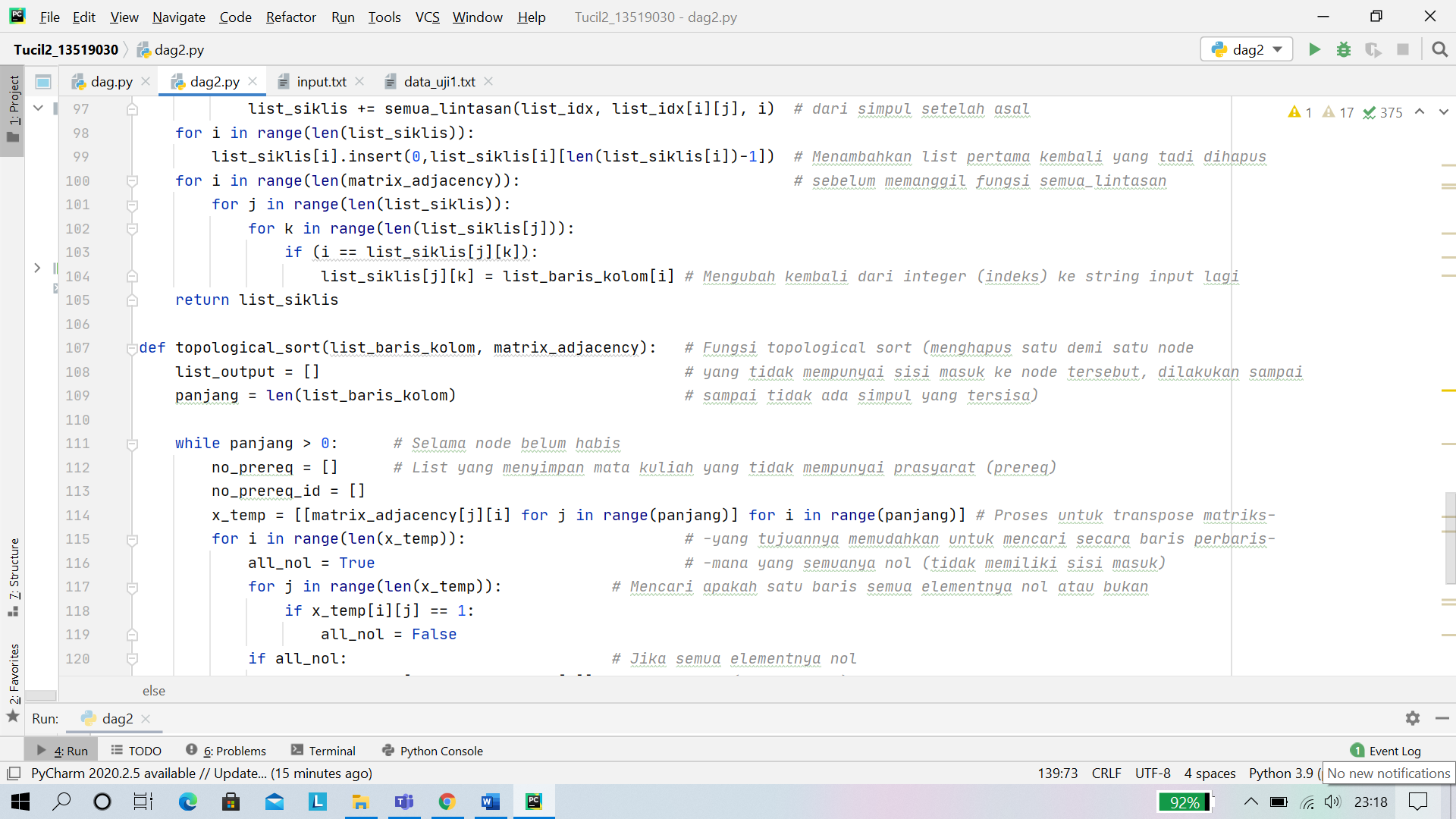


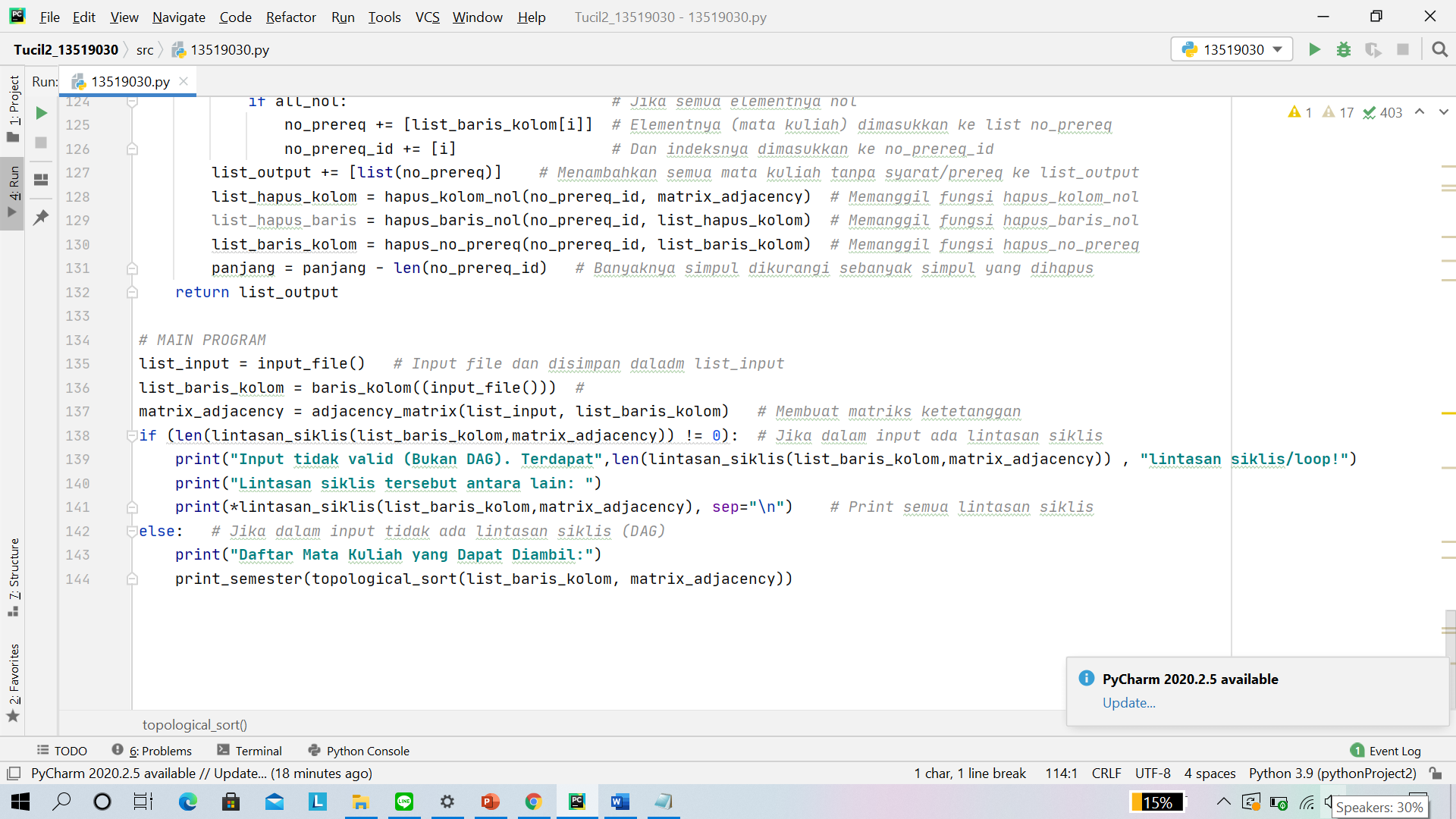












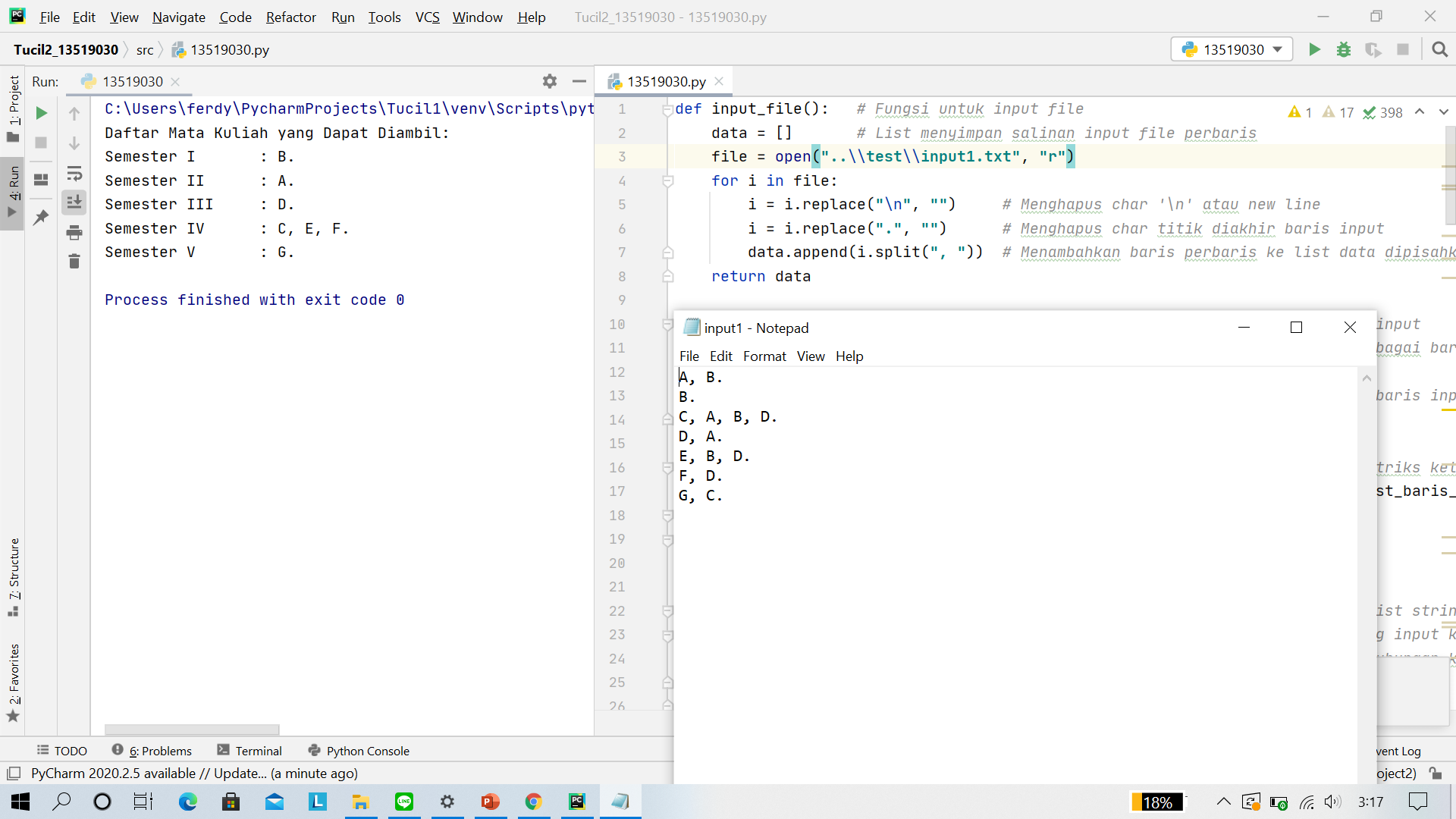
**2.2. *Copy Paste* (*Screenshot*) *Source Code***

def input\_file():data = [] file = open(**"input.txt"**, **"r"**)  
 for i in file:  
 i = i.replace(**"**\n**"**, **""**) i = i.replace(**"."**, **""**) data.append(i.split(**", "**)) return data  
  
def baris\_kolom(input\_list): baris\_kolom = [] for i in range(len(input\_list)):  
 baris\_kolom += [input\_list[i][0]] return baris\_kolom  
  
def adjacency\_matrix(list\_input, list\_baris\_kolom):  
 matrix = [[0 for i in range(len(list\_baris\_kolom))] for j in range(len(list\_baris\_kolom))]  
 for i in range(len(list\_input)):  
 for j in range(1,len(list\_input[i])):  
 k = 0  
 match = False  
 while k < len(list\_baris\_kolom) and match == False:  
 if list\_input[i][j] == list\_baris\_kolom[k]:  
 matrix[k][i] = 1  
 match = True  
 k += 1  
 return matrix  
  
def hapus\_kolom\_nol(list\_semua\_nol,matrix\_adjacency):  
 for i in range(len(list\_semua\_nol))[::-1]:  
 for j in matrix\_adjacency:  
 del j[list\_semua\_nol[i]]  
 return matrix\_adjacency  
  
def hapus\_baris\_nol(list\_semua\_nol,matrix\_adjacency):  
 for i in range(len(list\_semua\_nol))[::-1]:  
 del matrix\_adjacency[list\_semua\_nol[i]]  
 return matrix\_adjacency  
  
def hapus\_no\_prereq(no\_prereq, list\_baris\_kolom):  
 for i in range(len(no\_prereq))[::-1]:  
 del list\_baris\_kolom[no\_prereq[i]]  
 return list\_baris\_kolom  
  
def desimal\_ke\_romawi(desimal):  
 list\_desimal = [10, 9, 5, 4, 1]  
 list\_romawi = [**"X"**, **"IX"**, **"V"**, **"IV"**, **"I"**]  
 hasil = **''** i = 0  
 while desimal > 0:  
 for j in range(desimal // list\_desimal[i]):  
 hasil += list\_romawi[i]  
 desimal -= list\_desimal[i]  
 i += 1  
 return hasil  
  
def print\_semester(list\_output): for i in range(len(list\_output)):  
 if i+1 == 8: print(**"Semester"**, desimal\_ke\_romawi(i+1), end=**" : "**) elif i+1 == 3 or i+1 == 7: print(**"Semester"**, desimal\_ke\_romawi(i+1), end=**" : "**) elif i+1 == 1 or i+1 == 5: print(**"Semester"**, desimal\_ke\_romawi(i+1), end=**" : "**) else: print(**"Semester"**, desimal\_ke\_romawi(i+1), end=**" : "**) for j in range(len(list\_output[i])):  
 print(list\_output[i][j], end=**""**)  
 if j != len(list\_output[i]) - 1:  
 print(end=**", "**)  
 else:  
 print(**"."**)  
  
def semua\_lintasan(list\_input, asal, tujuan, list\_lintasan=[]):  
 list\_semua\_lintasan = []  
 list\_lintasan = list\_lintasan + [asal]  
 if asal == tujuan:  
 return [list\_lintasan]  
 if len(list\_input[asal]) == 0:  
 return []  
 for node in list\_input[asal]:  
 if node not in list\_lintasan:  
 list\_semua\_lintasan2 = semua\_lintasan(list\_input, node, tujuan, list\_lintasan)  
 for list\_lintasan2 in list\_semua\_lintasan2:  
 list\_semua\_lintasan += [list\_lintasan2]  
 return list\_semua\_lintasan  
  
def lintasan\_siklis(list\_baris\_kolom, matrix\_adjacency):  
 list\_idx = []  
 list\_siklis = []  
 for i in range(len(matrix\_adjacency)):  
 list\_idx\_temp = []  
 list\_idx\_temp += [list\_baris\_kolom[i]]  
 list\_idx += [list\_idx\_temp]  
 for j in range(len(matrix\_adjacency)):  
 if matrix\_adjacency[i][j] == 1:  
 list\_idx[i] += [j]  
 del list\_idx[i][0]  
 for i in range(len(list\_idx)):  
 for j in range(len(list\_idx[i])):  
 list\_siklis += semua\_lintasan(list\_idx, list\_idx[i][j], i)  
 for i in range(len(list\_siklis)):  
 list\_siklis[i].insert(0,list\_siklis[i][len(list\_siklis[i])-1])  
 for i in range(len(matrix\_adjacency)):  
 for j in range(len(list\_siklis)):  
 for k in range(len(list\_siklis[j])):  
 if (i == list\_siklis[j][k]):  
 list\_siklis[j][k] = list\_baris\_kolom[i]  
  
 return list\_siklis  
  
def topological\_sort(list\_baris\_kolom, matrix\_adjacency):  
 list\_output = []  
 panjang = len(list\_baris\_kolom)  
  
 while panjang > 0:  
 no\_prereq = []  
 no\_prereq\_id = []  
 x\_temp = [[matrix\_adjacency[j][i] for j in range(panjang)] for i in range(panjang)]  
 for i in range(len(x\_temp)):  
 all\_nol = True  
 for j in range(len(x\_temp)):  
 if x\_temp[i][j] == 1:  
 all\_nol = False  
 if all\_nol:  
 no\_prereq += [list\_baris\_kolom[i]]  
 no\_prereq\_id += [i]  
 list\_output += [list(no\_prereq)]  
 list\_hapus\_kolom = hapus\_kolom\_nol(no\_prereq\_id, matrix\_adjacency)  
 list\_hapus\_baris = hapus\_baris\_nol(no\_prereq\_id, list\_hapus\_kolom)  
 list\_baris\_kolom = hapus\_no\_prereq(no\_prereq\_id, list\_baris\_kolom)  
 panjang = panjang - len(no\_prereq\_id)  
 return list\_output  
  
*# MAIN PROGRAM*list\_input = input\_file()  
list\_baris\_kolom = baris\_kolom((input\_file()))  
matrix\_adjacency = adjacency\_matrix(list\_input, list\_baris\_kolom)  
if (len(lintasan\_siklis(list\_baris\_kolom,matrix\_adjacency)) != 0): print(**"Input tidak valid (Bukan DAG). Terdapat"**,len(lintasan\_siklis(list\_baris\_kolom,matrix\_adjacency)) , **"lintasan siklis/loop!"**)  
 print(**"Lintasan siklis tersebut antara lain: "**)  
 print(\*lintasan\_siklis(list\_baris\_kolom,matrix\_adjacency), sep=**"**\n**"**) else: print(**"Daftar Mata Kuliah yang Dapat Diambil:"**)  
 print\_semester(topological\_sort(list\_baris\_kolom, matrix\_adjacency))

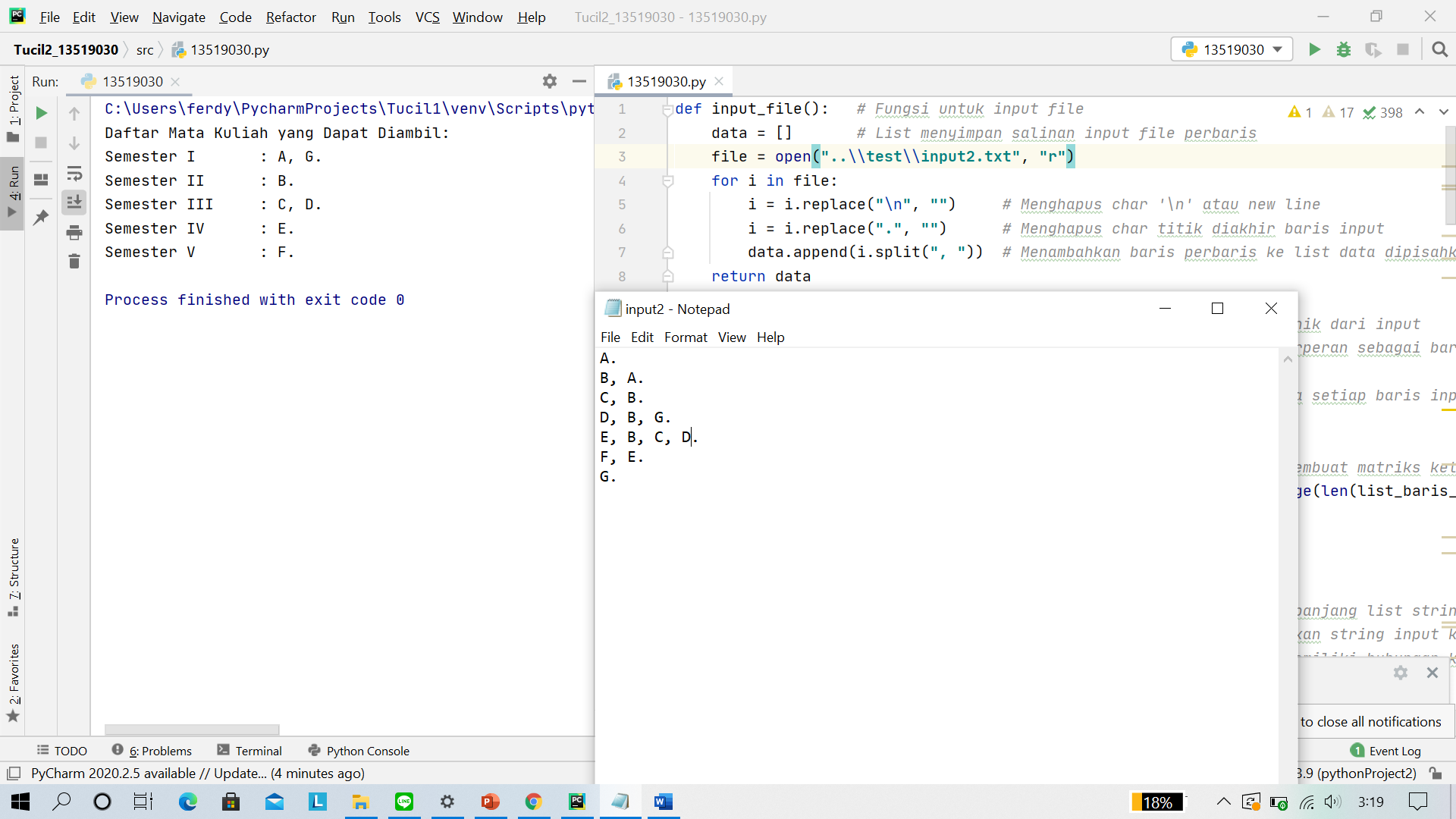
**BAB III**

**SCREENSHOT INPUT DAN OUTPUT**

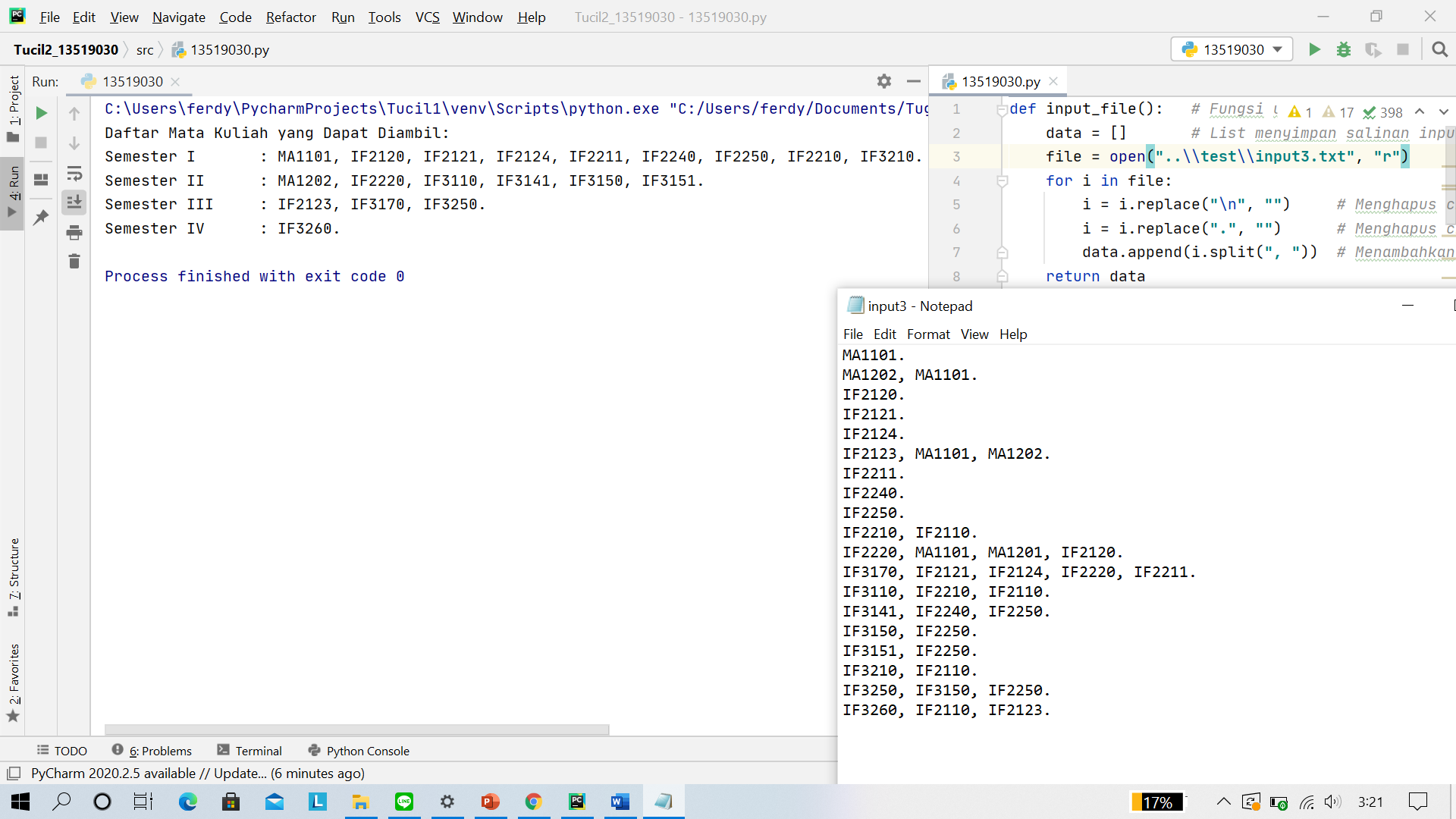
**3.1. Input ke-1**



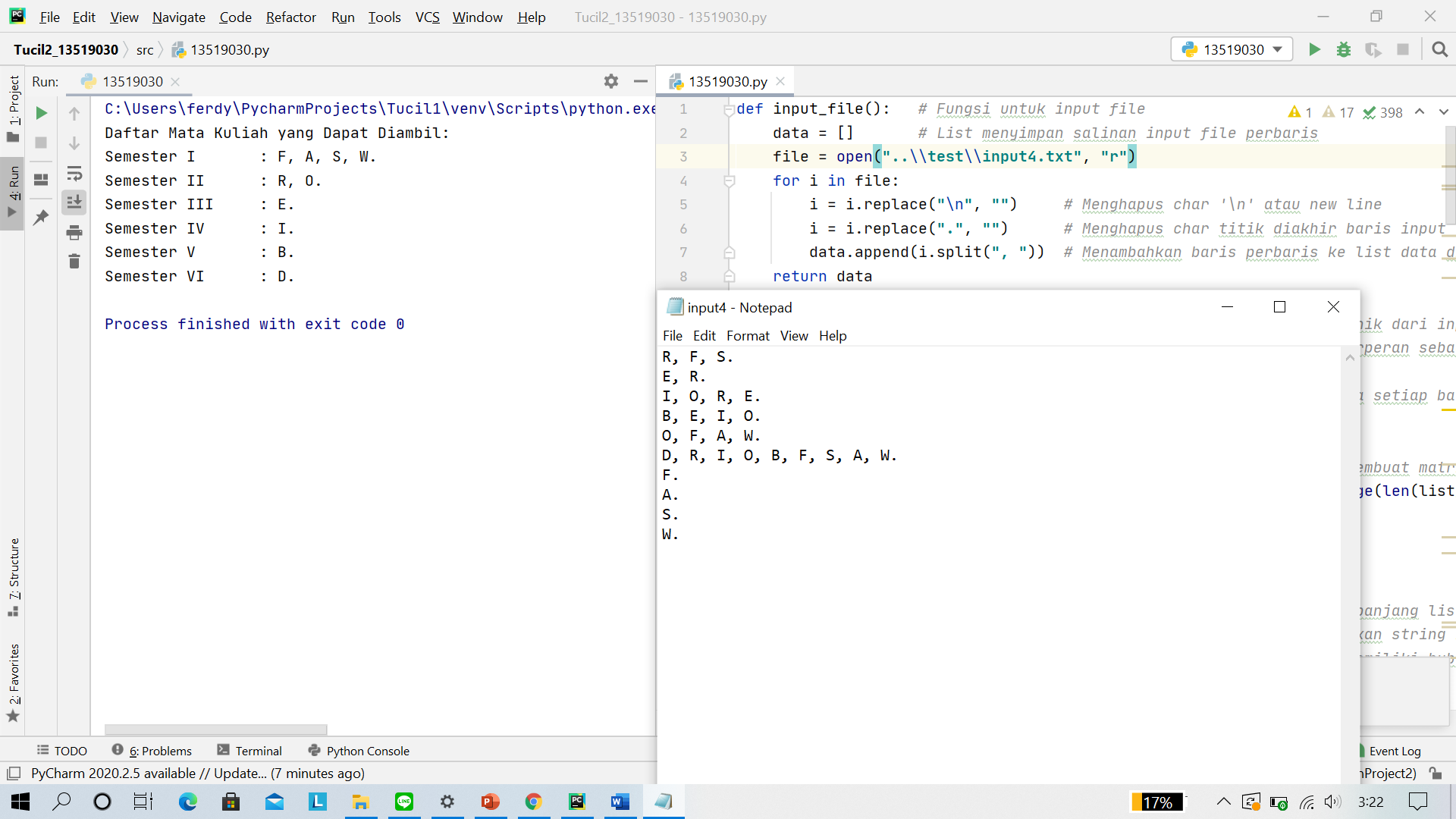
**3.2 Input ke-2**



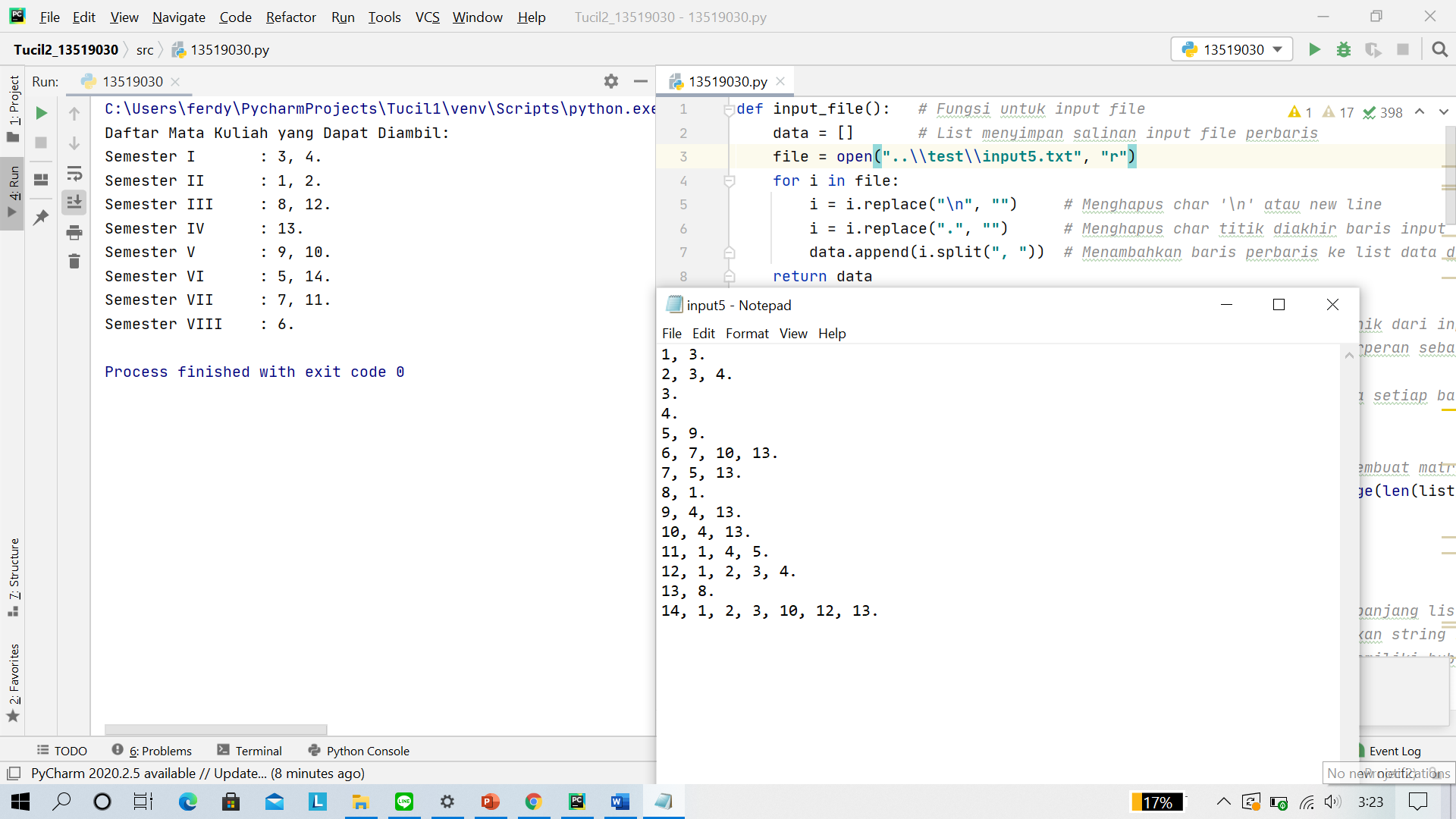
**3.3. Input ke-3**



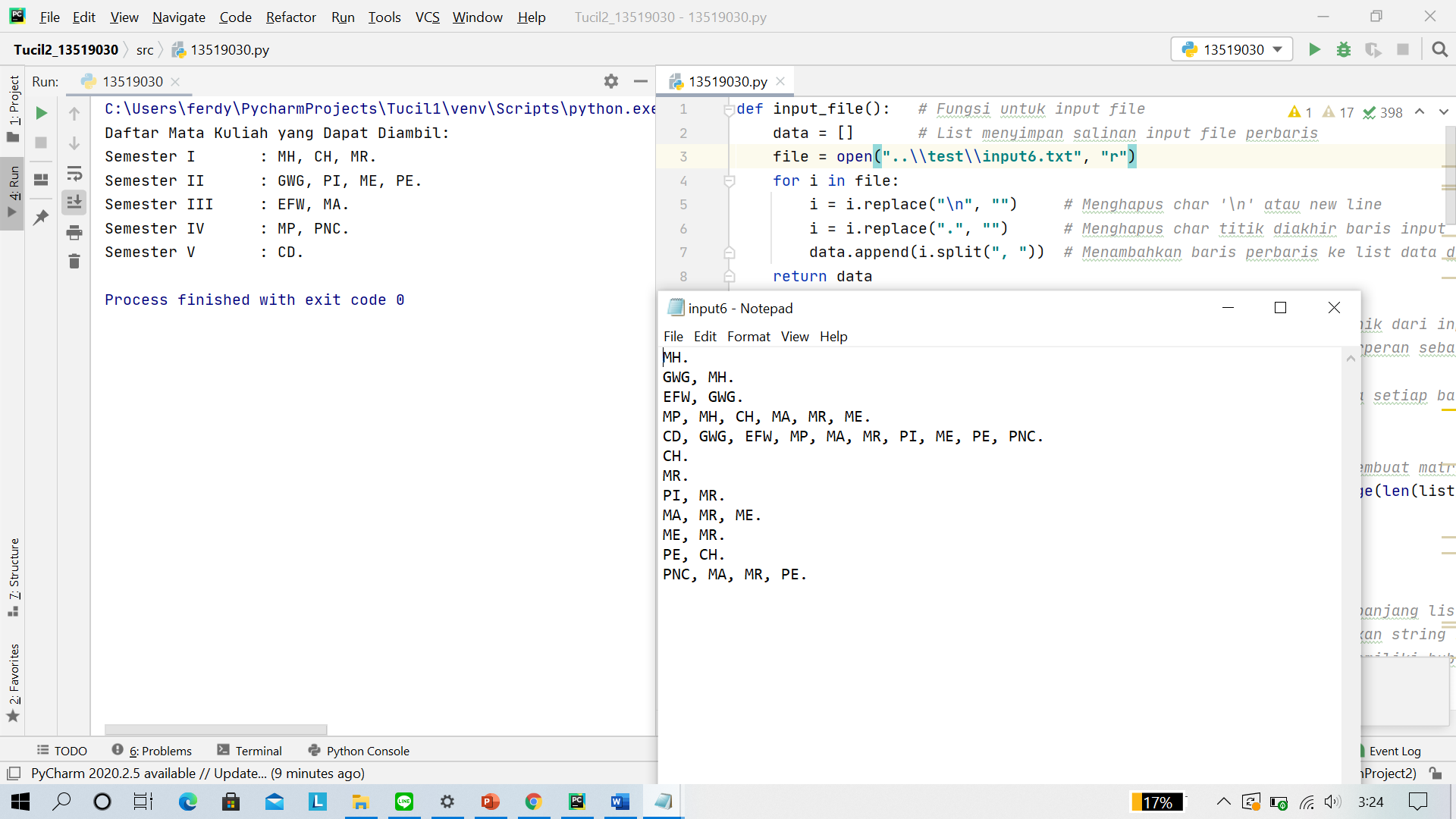
**3.4. Input ke-4**



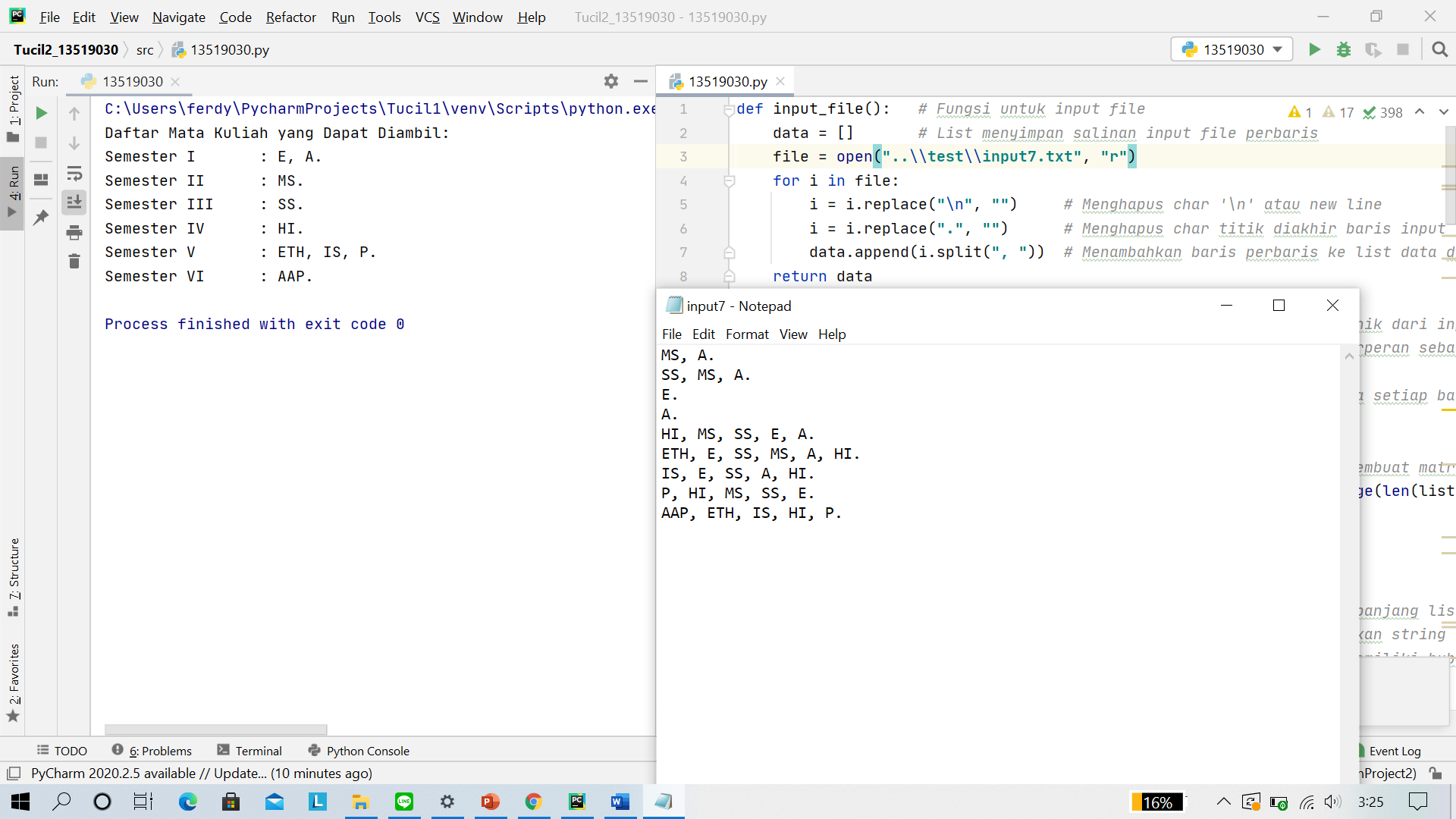
**3.5. Input ke-5**



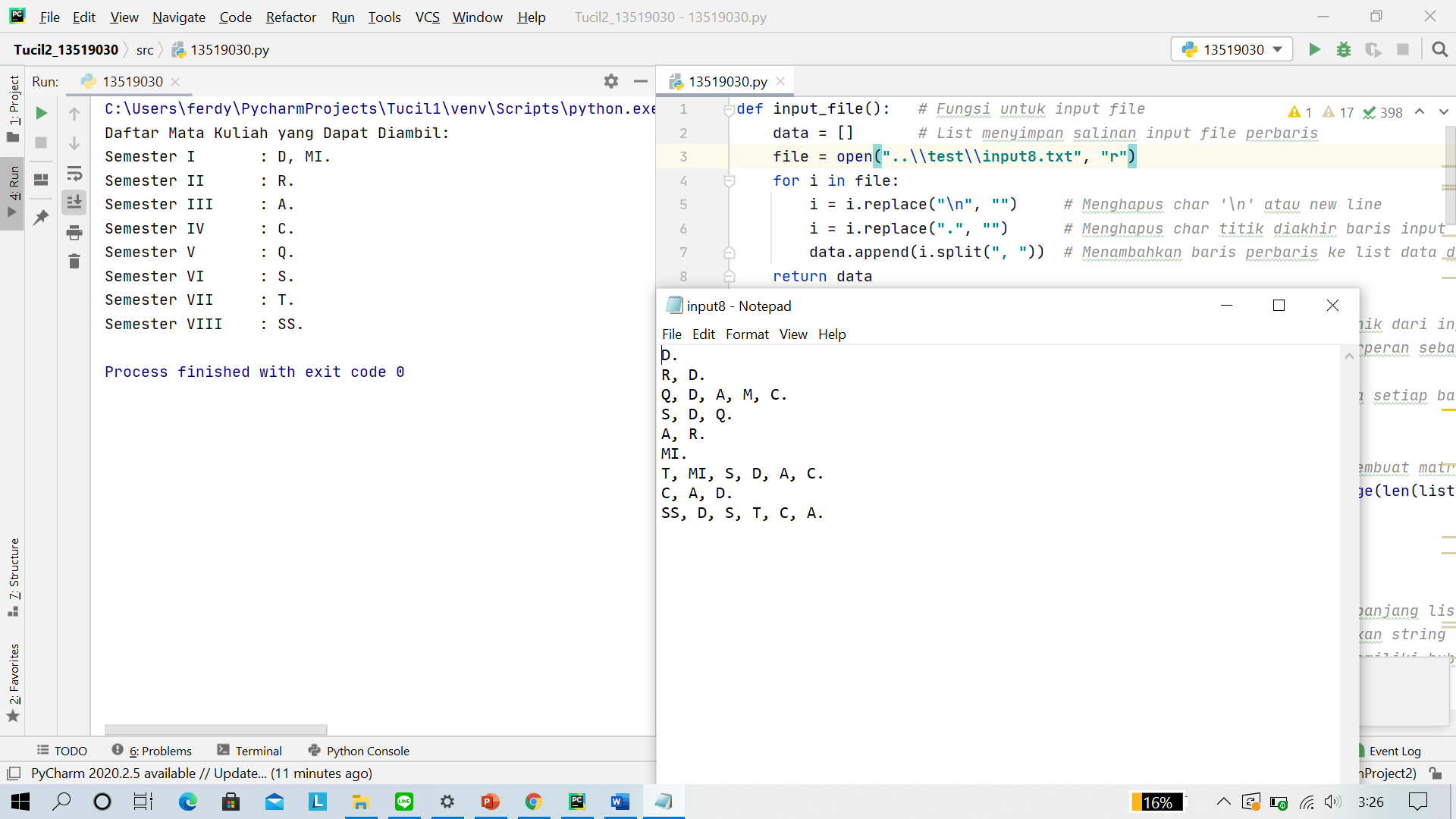
**3.6. Input ke-6**



**3.7. Input ke-7**

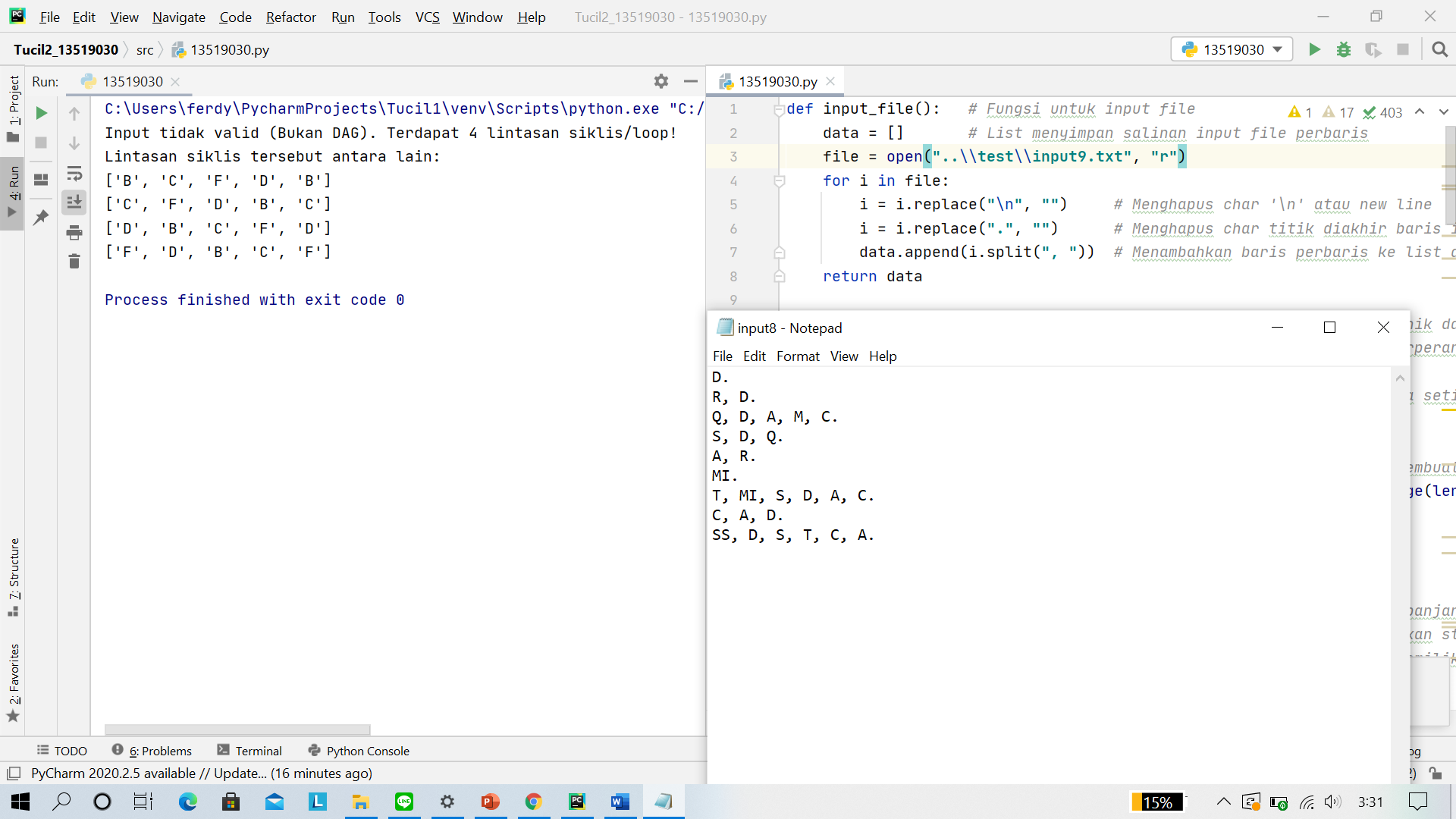


**3.8. Input ke-8**



**3.9. Input ke-9**

Karena dalam program saya ada kondisi untuk menangani jikalau input dari user buka DAG maka saya menyediakan satu input untuk contoh.



**BAB IV**

**ALAMAT REPOSITORY**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | ✓ |  |
| 1. Program berhasil *running* | ✓ |  |
| 1. Program dapat menerima berkas *input* dan menuliskan *output* | ✓ |  |
| 1. Luaran sudah benar untuk semua kasus *input*. | ✓ |  |