Tugas 15: Python Dasar (Bagian 3)

Nama: Bagus Alvin Gilang Pratama
NIM: 20220100118
Program Studi: Informatika

1. Buatkan kode program Python dari jawaban algoritma yang anda jawab pada tugas:
    1. Tugas 11: Studi Kasus Operasi Matriks (soal latihan 10.2 di halaman 83--84)
    2. Tugas 12: Studi Kasus Shortest Path Problem (soal latihan 11.2 di halaman 92.)
2. Buatkan jawabannya ke dalam bentuk PDF, kirim ke kolom File submissions.
3. Segera selesaikan agar dapat melanjutkan ke materi di minggu selanjutnya!

Jawaban: Tugas 11: Studi Kasus Operasi Matriks (soal latihan 10.2 di halaman 83--84)

```
M1 = [ [5, 2, 3], [1, 7, 1] ]
M2 = [ [1, 0], [1, 1], [2, 4] ]
print("M1 ", M1)
print("M2 ", M2)
M3 = {}
for i, d in enumerate(M1):
    M3[i] = {}
    for ii, dd in enumerate(M1[i]):
        for j, k in enumerate(M2):
            for jj, kk in enumerate(M2[j]):
                M3[i][jj] = 0
for i, d in enumerate(M1):
    for ii, dd in enumerate(M1[i]):
        for j, k in enumerate(M2):
            for jj, kk in enumerate(M2[j]):
                if(ii == j):
                    M3[i][jj] += M1[i][ii] * M2[j][jj]
                    print(M3[i][jj], '=', M1[i][ii], '*', M2[j][jj])
print("M3", M3)
M4 = []
for i in range(0, len(M3)):
    M4.append([])
    for ii in range(0, len(M3[i])):
        M4[i].append(M3[i][ii])
print("M4", M4)
```

Tugas 09: Tugas 12: Studi Kasus Shortest Path Problem (soal latihan 11.2 di halaman 92.)

```
def dijkstra(graph, start, end):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    previous_nodes = {node: None for node in graph}
    unvisited_nodes = graph.copy()

    while unvisited_nodes:
        current_node = min(unvisited_nodes, key=lambda x: distances[x])
        if distances[current_node] == float('inf'):
            break
        for neighbor, weight in graph[current_node].items():
            new_distance = distances[current_node] + weight
            if new_distance < distances[neighbor]:
                distances[neighbor] = new_distance
                previous_nodes[neighbor] = current_node
        unvisited_nodes.pop(current_node)

    path = []
    node = end
    while node != start:
        path.append(node)
        node = previous_nodes[node]
    path.append(start)
    path.reverse()

    return path, distances[end]

graph = {1: {2: 1, 3: 3},
         2: {3: 1, 5: 5},
         3: {1: 3, 4: 2},
         4: {5: 1},
         5: {}}

shortest_path, shortest_distance = dijkstra(graph, 1, 5)

print(shortest_path)
print(shortest_distance)
```