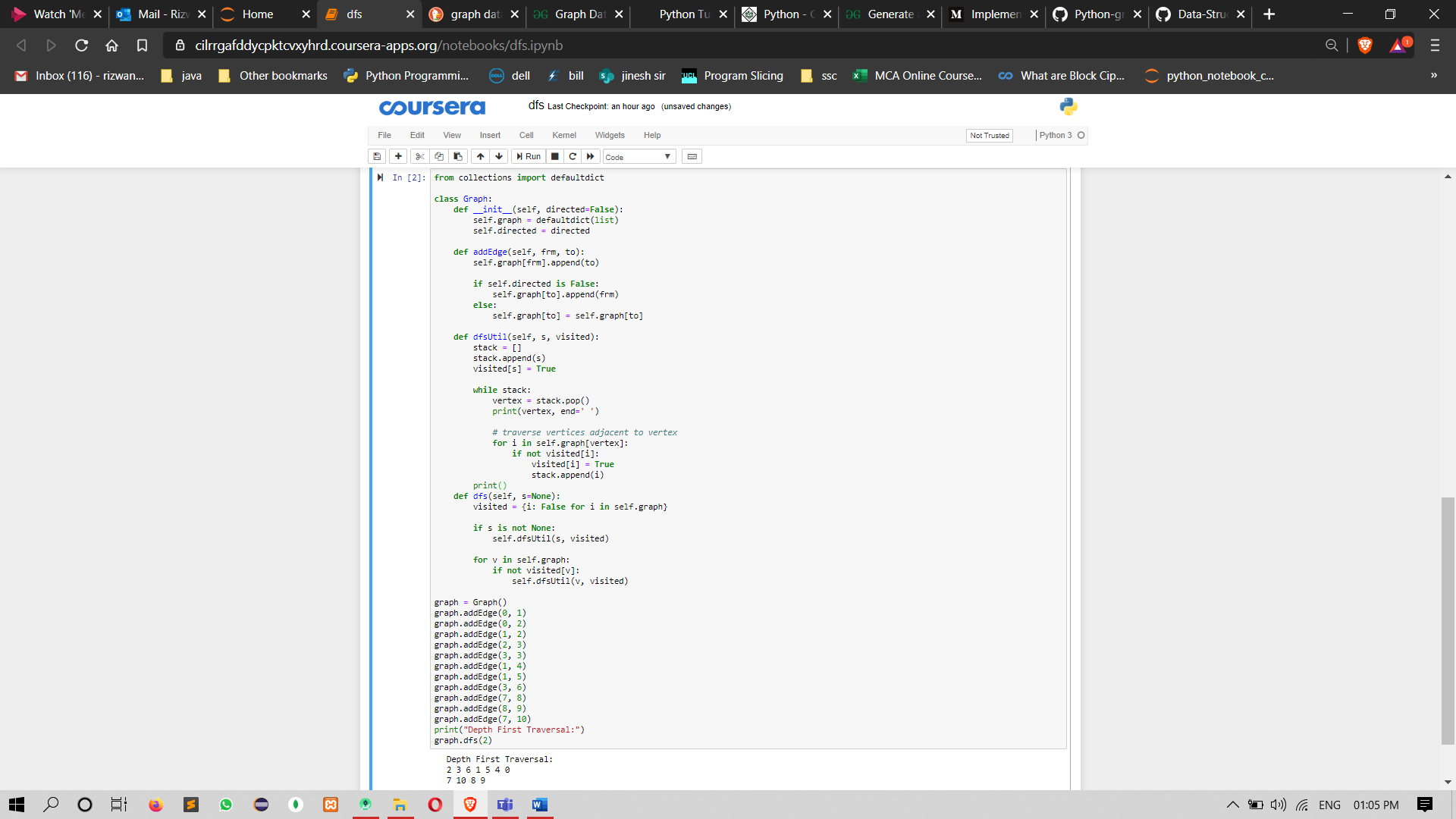
1. **DFS (17/Apr/2020)**
   1. **Store your graph using adjacency matrix(as we have done in the lab session)**
   2. **Write separate functions for the important operations(so that u can reuse those functions in the next programs too)**
   3. **DFS() should visit the nodes in the correct order and print them based on the increasing order of their Finishing time**



from collections import defaultdict

class Graph:

def \_\_init\_\_(self, directed=False):

self.graph = defaultdict(list)

self.directed = directed

def addEdge(self, frm, to):

self.graph[frm].append(to)

if self.directed is False:

self.graph[to].append(frm)

else:

self.graph[to] = self.graph[to]

def dfsUtil(self, s, visited):

stack = []

stack.append(s)

visited[s] = True

while stack:

vertex = stack.pop()

print(vertex, end=' ')

# traverse vertices adjacent to vertex

for i in self.graph[vertex]:

if not visited[i]:

visited[i] = True

stack.append(i)

print()

def dfs(self, s=None):

visited = {i: False for i in self.graph}

if s is not None:

self.dfsUtil(s, visited)

for v in self.graph:

if not visited[v]:

self.dfsUtil(v, visited)

graph = Graph()

graph.addEdge(0, 1)

graph.addEdge(0, 2)

graph.addEdge(1, 2)

graph.addEdge(2, 3)

graph.addEdge(3, 3)

graph.addEdge(1, 4)

graph.addEdge(1, 5)

graph.addEdge(3, 6)

graph.addEdge(7, 8)

graph.addEdge(8, 9)

graph.addEdge(7, 10)

print("Depth First Traversal:")

graph.dfs(2)//sorce node =2