ModelingFunctions

June 19, 2022

[4]:
```
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'

# print(color.PURPLE + 'Hello World!' + color.END)
```

[5]:
```
def explain():
    print(color.UNDERLINE + color.BOLD + 'IMPORTED FUNCTIONS:\n\n' + color.END)
    print(color.BOLD + 'times(start, stop): ' + color.END + 'creates list of␣
↪times to record data for input has all nodes and their edges\n\n')
    print(color.BOLD + 'edgeslist(input1): ' + color.END + 'outputs the edge␣
↪labels as seen in the Labeled Map of Knoxville\n\n')
    print(color.BOLD + 'traffic(times, edges): ' + color.END + 'allows user to␣
↪input traffic conditions and outputs as a list\n\n')
    print(color.BOLD + 'trafficTxt(times, edges, filename="TrafficConditions.
↪txt", accessmode="w+"): ' + color.END + 'saves as file\n\n')
    print(color.BOLD + 'timeEachNode (distances, trafficConditions, timeOfDay,␣
↪Day="Sun"): ' + color.END + 'outputs a list of weights of each edge\n\n')
    print(color.BOLD + 'timeEachNodeTxt (distances, trafficConditions,␣
↪timeOfDay, Day="Sun", filename = "NodeWeights.txt", \naccessmode = "w+"): '␣
↪+ color.END + 'saves as file\n\n')
    print(color.BOLD + 'timeEachTxt (input1, input2, distances,␣
↪trafficConditions, timeOfDay, Day="Sun", filename = "EdgeWeights.txt", \n␣
↪accessmode = "w+"): ' + color.END + 'saves as file\n\n')
    print(color.BOLD + 'onebigfunction(timeOfDay, Day, trafficConditions,␣
↪filename="EdgeWeights.txt", \n accessmode = "w+"): ' + color.END + 'saves as␣
↪file\n\n')
    print(color.BOLD + 'color Class: ' + color.END + 'purple, cyan, darkcyan,␣
↪blue, green, yellow, red, bold, underline, end')
```

```
explain()
```

IMPORTED FUNCTIONS:
times(start, stop): creates list of times to record data for input has all nodes and their edges

edgeslist(input1): outputs the edge labels as seen in the Labeled Map of Knoxville

traffic(times, edges): allows user to input traffic conditions and outputs as a list

trafficTxt(times, edges, filename="TrafficConditions.txt", accessmode="w+"):

saves as file

timeEachNode (distances, trafficConditions, timeOfDay, Day="Sun"):

outputs a list of weights of each edge

timeEachNodeTxt (distances, trafficConditions, timeOfDay, Day="Sun",

filename = "NodeWeights.txt",

accessmode = "w+"): saves as file

timeEachTxt (input1, input2, distances, trafficConditions, timeOfDay,

Day="Sun", filename = "EdgeWeights.txt",

accessmode = "w+"): saves as file

onebigfunction(timeOfDay, Day, trafficConditions,

filename="EdgeWeights.txt",

accessmode = "w+"): saves as file

color Class: purple, cyan, darkcyan, blue, green, yellow, red, bold, underline, end

[3]:
```
import math
```

0.0.1 Times Function

creates a vector of the times I get data from traffic simulator. See example output.

[4]:
```
def times(start, stop):
    times = []
    ran = int( 4*(stop - start)/100 + 1 )
    for i in range(0, ran):
        if (start % 100) == 60:
            start += 40
        times.append(start)
        start += 15
    return times
```

[5]:
```
#creates vector of times from 6:00 am to 7:00 am
extimes = times(600, 700)
```

[6]:
```
# edges with the nodes
input1 = [('1', 'A', 'B'), ('2', 'B', 'D'), ('3', 'B', 'E'), ('4', 'A', 'C'),␣
 ↪('5', 'D', 'M'), ('6', 'C', 'F'), ('7', 'F', 'G'), ('8', 'G', 'H'), ('9',␣
 ↪'H', 'I'), ('10', 'I', 'J'), ('11', 'I', 'K'), ('12', 'J', 'L'), ('13', 'D',␣
 ↪'K'), ('14', 'K', 'L'), ('15', 'L', 'M'), ('16', 'M', 'N'), ('17', 'E',␣
 ↪'N'), ('18', 'N', 'O'), ('19', 'D', 'F'), ('20', 'G', 'K')]

# nodes with the edges
input2 = [('A', '1', '4'), ('B', '1', '2', '3'), ('C', '4', '6'), ('D', '2',␣
 ↪'5', '13', '19'), ('E', '3', '17'), ('F', '6', '7'), ('G', '7', '8'), ('H',␣
 ↪'8', '9'), ('I', '9', '10', '11'), ('J', '10', '12'), ('K', '11', '13',␣
 ↪'14', '20'), ('L', '12', '14', '15'), ('M', '5', '15', '16'), ('N', '16',␣
 ↪'17', '18'), ('O', '18')]
```

0.0.2 Edges Function

given a list of nodes with each possible edge, prints out the edge labels as seen in the Labeled Map of Knoxville below

[7]:
```
def edgeslist(input1):
    out=[]
    length = len(input1)
    for i in range(0, 2*length):
        out.append(input1[math.floor(i/2)][0]+input1[math.floor(i/2)][(i%2) +␣
 ↪1])
    return out
```

[8]:
```
def edgeslist2(input2):
    out = []
    for i in range(len(input2)):
        for j in range(1, len(input2[i])):
            out.append(input2[i][j]+input2[i][0])
    return out
```

[9]:
```
edgeslist2(input2)
```

[9]:
```
['1A',
 '4A',
 '1B',
 '2B',
 '3B',
 '4C',
 '6C',
 '2D',
 '5D',
 '13D',
 '19D',
 '3E',
 '17E',
 '6F',
 '7F',
 '7G',
 '8G',
 '8H',
 '9H',
 '9I',
 '10I',
 '11I',
 '10J',
 '12J',
 '11K',
 '13K',
 '14K',
 '20K',
 '12L',
 '14L',
 '15L',
 '5M',
 '15M',
 '16M',
 '16N',
 '17N',
 '18N',
 '18O']
```

[10]:
```
exinput = [('1', 'A', 'B'), ('2', 'B', 'C'), ('3', 'A', 'C')]
edgeslist(exinput)
```

[10]:
```
['1A', '1B', '2B', '2C', '3A', '3C']
```

[1]:
```
#![Hodges%20Neyland.jpg](attachment:Hodges%20Neyland.jpg)
```

0.0.3 Traffic Conditions Function

given the times and edges from the previous 2 functions, allows the user to input the traffic conditions for each edge, and saves and outputs those conditions into a list.

[11]:
```
def traffic(times, edges):
    output = []
    for j in range(len(times)):
        print("You are entering conditions for {0}. Or enter 'Q' to quit. Your␣
↪progress will be saved and returned.".format(times[j]))
        for i in range(len(edges)):
            condition = input("Enter the traffic condition for edge "+␣
↪edges[i]+ ": ")
            while condition != "G" and condition != "O" and condition!="R" and␣
↪condition!="B" and condition!="NA":
                if condition == "Q":
                    print("Quitting program.\n")
                    print("Calculating weights...")
                    return output
                print("Invalid input. Enter G for green, O for orange, R for␣
↪red, B for brown, and NA if not available.")
                condition = input("\n Enter the traffic condition for edge " +␣
↪edges[i]+ ": ")
            element = [times[j], edges[i], condition]
            output.append(element)
    print("Calculating weights...\n")
    return output
```

[12]:
```
def trafficTxt (times, edges, filename="TrafficConditions.txt",␣
↪accessmode="w+"):
    file = open(filename, accessmode)
    for j in range(len(times)):
        print("You are entering conditions for {0}. Or enter 'Q' to quit. Your␣
↪progress will be saved.".format(times[j]))
        for i in range(len(edges)):
            condition = input("Enter the traffic condition for edge "+␣
↪edges[i]+ ": ")
            condition = condition.upper()
            while condition != "G" and condition != "O" and condition!="R" and␣
↪condition!="B" and condition!="NA":
                if condition == "Q":
                    print("Quitting program.\n")
                    file.close()
                    return
                print("Invalid input. Enter G for green, O for orange, R for␣
↪red, B for brown, and NA if not available.")
                condition = input("\n Enter the traffic condition for edge " +␣
↪edges[i]+ ": ")
```

```
            element = "%s %s %s \n" %(times[j], edges[i], condition)
            file.write(element)
    file.close()
```

[2]:
```
## list of all distances of edges in feet

distancesDict = dict([('1A', 377), ('1B', 377), ('2B', 528), ('2D', 528),␣
↪('3B', 1056), ('3E', 1056), ('4A', 367), ('5D', 528), ('5M', 528), ('6C',␣
↪430), ('6F', 430), ('7F', 400), ('7G', 400), ('8G', 302), ('8H', 302),␣
↪('9H', 528), ('9I', 528), ('10I', 528), ('10J', 528), ('11I', 300), ('11K',␣
↪300), ('12J', 318), ('12L', 318), ('13K', 350), ('13D', 350), ('14K', 528),␣
↪('15L', 400), ('15M', 400), ('16M', 2112), ('16N', 2112), ('17E', 528),␣
↪('17N', 528), ('18N', 528), ('18O', 528), ('19D', 528), ('19F', 528),␣
↪('20K', 528)])
distancesList = [('1A', 377), ('1B', 377), ('2B', 528), ('2D', 528), ('3B',␣
↪1056), ('3E', 1056), ('4A', 367), ('5D', 528), ('5M', 528), ('6C', 430),␣
↪('6F', 430), ('7F', 400), ('7G', 400), ('8G', 302), ('8H', 302), ('9H',␣
↪528), ('9I', 528), ('10I', 528), ('10J', 528), ('11I', 300), ('11K', 300),␣
↪('12J', 318), ('12L', 318), ('13K', 350), ('13D', 350), ('14K', 528),␣
↪('15L', 400), ('15M', 400), ('16M', 2112), ('16N', 2112), ('17E', 528),␣
↪('17N', 528), ('18N', 528), ('18O', 528), ('19D', 528), ('19F', 528),␣
↪('20K', 528)]
```

[3]:
```
distancesList[0]
```

[3]:
```
('1A', 377)
```

0.0.4 timeEachNode Function

Outputs a list of edges and their travel times, given the information from the traffic function.

[14]:
```
def timeEachNode (distances, trafficConditions, timeOfDay, Day='Sun'):
    length = len(distances)
    travelTimes = []
    test = []
    ctr = 0
    for i in range(length):
        edge = distances[i][0]
        dist = distances[i][1]
        mint = timeOfDay % 100
        index = int (mint / 15 + ( (timeOfDay - mint) - 600 ) / 25 * 40 ) + ctr
        thisCond = trafficConditions[index][2]
        test.append([index, thisCond])
        Time = dist / 44
        traffic = trafficFunction(thisCond, Time)
        thisTravelTime = '{0}'.format(Time + traffic)
            # speed limit in Knoxville ~ 30 mph is 44 ft/s
```

```
        travelTimes.append([edge, thisTravelTime])
        ctr += 1
    return(travelTimes)
```

[15]:
```
def timeEachNodeTxt (distances, trafficConditions, timeOfDay, Day='Sun',␣
↪filename = "NodeWeights.txt", accessmode = "w+"):
    length = len(distances)
    ctr = 0
    file = open(filename, accessmode)
    for i in range(length):
        edge = distances[i][0]
        dist = distances[i][1]
        mint = timeOfDay % 100
        index = int (mint / 15 + ( (timeOfDay - mint) - 600 ) / 25 * 40 ) + ctr
        thisCond = trafficConditions[index][2]
        Time = dist / 44
            # speed limit in Knoxville ~ 30 mph is 44 ft/s
        traffic = trafficFunction(thisCond, Time)
        thisTravelTime = '{0}'.format(Time + traffic)
        file.write("%s %s\n" %(edge, thisTravelTime))
        ctr += 1
```

[1]:
```
def timeEachTxt (input1, input2, distances, trafficConditions, timeOfDay,␣
↪Day='Sun', filename="EdgeWeights.txt", accessmode = "w+"):
    length = len(distances)
    file = open(filename, accessmode)
    file.write("%s\n" %(len(input2)))
    minute = timeOfDay % 100
    index2 = int (minute / 15 + ( (timeOfDay - minute) - 600 ) / 25 * 40 )
    for i in range(length):
        edgen = distances[i][0]
        j = 0
        while edgen[j].isnumeric():
            edge = edgen[:j+1]
            j+=1
        k = 1
        node1 = edgen[k:] #from
        if not(node1.isalpha()):
            k+=1
            node1 = edgen[k:]
        index1 = int(edge) - 1
        node2 = input1[index1][1] #to
        if node2 == node1:
            node2 = input1[index1][2]
        dist = distances[i][1]
        index2 += 1
        thisCond = trafficConditions[index2][2]
```

```
        Time = dist / 44
            # speed limit in Knoxville ~ 30 mph is 44 ft/s
        traffic = trafficFunction(thisCond, Time)
        thisTravelTime = '{0}'.format(Time + traffic)
        file.write("%s %s %s\n" %(node1, node2, thisTravelTime))
```

[14]:
```
def trafficFunction(trafficCond, Time):
    if trafficCond == 'G':
        traffic = 1.1*Time
    elif trafficCond == 'O':
        traffic = 1.25*Time
    elif trafficCond == 'R':
        traffic = 2*Time
    elif trafficCond == 'B':
        traffic = 3*Time
    elif trafficCond == 'NA':
        traffic = 0
    return traffic
# These are test numbers to see if the function works.
# I will study the google maps times later to get more accurate functions
```

[5]:
```
def onebigfunction(timeOfDay, Day, trafficConditions, filename='EdgeWeights.
 ↪txt', accessmode='w+'):
    distances = [('1A', 377), ('1B', 377), ('2B', 528), ('2D', 528), ('3B',␣
 ↪1056), ('3E', 1056), ('4A', 367), ('5D', 528), ('5M', 528), ('6C', 430),␣
 ↪('6F', 430), ('7F', 400), ('7G', 400), ('8G', 302), ('8H', 302), ('9H',␣
 ↪528), ('9I', 528), ('10I', 528), ('10J', 528), ('11I', 300), ('11K', 300),␣
 ↪('12J', 318), ('12L', 318), ('13K', 350), ('13D', 350), ('14K', 528),␣
 ↪('15L', 400), ('15M', 400), ('16M', 2112), ('16N', 2112), ('17E', 528),␣
 ↪('17N', 528), ('18N', 528), ('18O', 528), ('19D', 528), ('19F', 528),␣
 ↪('20K', 528)]
    input1 = [('1', 'A', 'B'), ('2', 'B', 'D'), ('3', 'B', 'E'), ('4', 'A',␣
 ↪'C'), ('5', 'D', 'M'), ('6', 'C', 'F'), ('7', 'F', 'G'), ('8', 'G', 'H'),␣
 ↪('9', 'H', 'I'), ('10', 'I', 'J'), ('11', 'I', 'K'), ('12', 'J', 'L'),␣
 ↪('13', 'D', 'K'), ('14', 'K', 'L'), ('15', 'L', 'M'), ('16', 'M', 'N'),␣
 ↪('17', 'E', 'N'), ('18', 'N', 'O'), ('19', 'D', 'F'), ('20', 'G', 'K')]
    input2 = [('A', '1', '4'), ('B', '1', '2', '3'), ('C', '4', '6'), ('D',␣
 ↪'2', '5', '13', '19'), ('E', '3', '17'), ('F', '6', '7'), ('G', '7', '8'),␣
 ↪('H', '8', '9'), ('I', '9', '10', '11'), ('J', '10', '12'), ('K', '11',␣
 ↪'13', '14', '20'), ('L', '12', '14', '15'), ('M', '5', '15', '16'), ('N',␣
 ↪'16', '17', '18'), ('O', '18')]
    length = len(distances)
    file = open(filename, accessmode)
    file.write("%s\n" %(len(input2)))
    minute = timeOfDay % 100
    index2 = int (minute / 15 + ( (timeOfDay - minute) - 600 ) / 25 * 40 )
    for i in range(length):
```

```
        edgen = distances[i][0]
        j = 0
        while edgen[j].isnumeric():
            edge = edgen[:j+1]
            j+=1
        k = 1
        node1 = edgen[k:] #from
        if not(node1.isalpha()):
            k+=1
            node1 = edgen[k:]
        index1 = int(edge) - 1
        node2 = input1[index1][1] #to
        if node2 == node1:
            node2 = input1[index1][2]
        dist = distances[i][1]
        index2 += 1
        thisCond = trafficConditions[index2][2]
        Time = dist / 44
            # speed limit in Knoxville ~ 30 mph is 44 ft/s
        traffic = trafficFunction(thisCond, Time)
        thisTravelTime = '{0}'.format(Time + traffic)
        file.write("%s %s %s\n" %(node1, node2, thisTravelTime))
        ## need to change function when we have more than one day.
```