Programa DELL IT Academy - Processo Seletivo - Turma 12
Autor do código: Paulo Renato Ficks Júnior

LOGÍSTICA PELO TERRITÓRIO BRASILEIRO

Para desenvolver o problema de logística foi utilizado o Python, versão 3.9.
Entre as três linguagens que eu trabalhei (C + +, Java e Python) o Python parece ter uma manipulação melhor de dados em .CSV.

Primeiro bloco

O código fonte foi realizado em etapas. na primeira etapa é cabeçalho onde são feitas as declarações de 'list' uma delas ("tabela_cidades_distancia") será utilizada para armazenar os dados da tabela e a outra 'list' ("rota") serve para armazenamento temporário das cidades da rota escolhida no MENU #3. Há ainda a declaração de duas variáveis simples: o "soma3" e o "custoKM". A primeira é para armazenar a soma da distância entre as cidades de uma rota escolhida no MENU#3. A segunda é para armazenar o custo por quilômetro que será utilizado no MENU#1

```python
# código fonte versão 3.9 Python

import csv  # importando a biblioteca csv para trabalhar este formato no python.

tabela_cidades_distancia = []  # declarando a list 'tabela_cidades_distancia' que constará as cidades e suas
# respectivas distâncias.

with open("DNIT-Distancias.csv") as csvfile:
    reader = csv.reader(csvfile, delimiter=";", quoting=csv.QUOTE_ALL)

    # transformando linhas em 'list'.
    for row in reader:
        tabela_cidades_distancia.append(row)

rota = []  # declarando a list 'rota' que constará as rotas entre cidades escolhidas no item #3 do MENU.

soma3 = 0  # declarando a variavel uniforme para o somatório de kilometragem das rotas do item #3 do MENU.

custoKM = 0

# início do loop
```

Imagem 1: cabeçalho inicial do programa

Ainda no cabeçalho, foi utilizada a biblioteca CSV para que pudesse ser manipulada a tabela fornecida em formato .csv. Com o comando de FOR iniciais é transformado o modelo .csv em uma lista múltipla dentro do Python, para que cada elemento ocupasse um espaço distinto e único no código, com finalidade de utilizar o número de posição de cada elemento e subelemento da lista de cidades/distância como forma similar a uma matriz.

Segundo bloco

Na segunda parte é iniciado o Loop. É necessário colocar o MENU como uma forma de Loop com finalidade de retornar ao início do programa, após entrada nos itens do MENU, sem ter que reiniciá-lo com consequência, no Python, de perder os dados imputados.

```
21  # início do loop
22  while True:
23      print('=' * 5, 'ESCOLHA UMA OPÇÃO', '=' * 5)
24      print('1. Configurar custo por KM')
25      print('2. Consultar trecho')
26      print('3. Consultar rota')
27      print('4. Terminar o programa\n')
28
29      # novo loop para que o valor digitado no menu inicial seja conforme solicitado segundo opções.
30      while True:
31          try:
32              escolha = int(input('Digite a opção aqui: '))
33              if escolha not in (1, 2, 3, 4):
34                  raise ValueError('Valor fora do limite permitido')
35          except ValueError as e:
36              print('Valor inválido: Utilizar somente números de 1 a 4.')
37          else:
38              break
39
40      # configurando escolha de MENU #1 para vir valores de custo positivos e formatos corretos.
```

imagem 2: Início do Loop do MENU

Na segunda parte, também, é “printada” na tela o MENU do programa de logística, onde é pedido que se faça a escolha entre os itens do MENU. Há uma restrição imposta ao usuário caso ele não escolha um dos números do item como: utilizar números maiores ou menores que o permitido ou sequer digitar um caractere inválido. O código retorna à escolha de um item do MENU até ser definido um número correto, sem apresentar erros ou bugs.

Terceiro bloco

No terceiro bloco do programa é a parte que inicia caso o usuário escolha o item #1 do MENU (definição do custo por KM da viagem). Neste bloco é definido o curso por quilômetro da viagem. Há uma restrição imposta ao usuário caso ele determine um valor nulo ou negativo para o custo por KM ou sequer digitar um número, que pode ser do tipo float ou integer. Caso ele faça a escolha errada é iniciado um loop até que ele escolha um número maior do que zero.

```
40      # configurando escolha de MENU #1 para vir valores de custo positivos e formatos corretos.
41      if escolha == 1:
42          while True:
43              try:
44                  custoKM = float(input('Informe o custo por KM rodado, em R$: '))
45                  if custoKM <= 0:
46                      raise ValueError('Valor inválido: Utilizar somente números e valores positivos não nulos.')
47              except ValueError as e:
48                  print('Valor inválido: Utilizar somente números e valores positivos não nulos.')
49              else:
50                  break
51
```

imagem 3: item #1 do MENU

O custo por quilômetro é armazenada na variável simples custoKM que será utilizada nos itens #2 e #3 do MENU para o custo final em R$ de cada rota/trecho.

Quarto bloco

Na quarta etapa é a parte do programa que inicia quando o usuário escolhe o item #2 do MENU (trecho entre 2 cidades). Já no início é imposta uma restrição ao usuário que informe inicialmente um custo por KM, ou seja, que o usuário escolha o item #1 do MENU e defina o custo, antes de escolher os outros itens do MENU, pois no item #2 é feito o cálculo do custo do trecho, necessitando o valor do curso por KM.

Para o quarto bloco é pedido ao usuário o nome da cidade de origem e o da cidade de destino. Há uma restrição ao usuário caso ele não digite corretamente o nome da cidade, seja por erro de digitação ou por a cidade não constar na lista de viagem. É iniciado um loop em cada escolha de cidade (origem e destino) até que o usuário defina corretamente o nome da cidade.

```
# configurando escolha de MENU #2 para um trecho entre 2 cidades.
if escolha == 2:
    if custoKM == 0:
        print('Forneça o valor por KM da viagem no item 1 do MENU.\n')
        continue
    origem = str(input('Informe a Cidade de Origem: ').strip().upper())
    destino = str(input('Informe a Cidade de Destino: ').strip().upper())

    # condição caso a cidade de origem não conste na tabela_cidades_distancia, com repetição.
    while origem not in tabela_cidades_distancia[0]:
        print('Erro de digitação, favor não utilizar acentos ou números.')
        origem = str(input('Informe a cidade de origem novamente: ').strip().upper())
    pos_o = tabela_cidades_distancia[0].index(origem)

    # condição caso a cidade de destino não conste na tabela_cidades_distancia, com repetição.
    while destino not in tabela_cidades_distancia[0]:
        print('Erro de digitação, favor não utilizar acentos ou números.')
        destino = str(input('Informe a cidade de destino novamente: ').strip().upper())
    pos_d = tabela_cidades_distancia[0].index(destino)

    # através dos índices da matriz criada 'tabela_cidades_distancia' determinar a distância as cidades do trecho.
    distancia = int(tabela_cidades_distancia[pos_o + 1][pos_d])
    print(f'O custo da Viagem é R$ {custoKM * distancia:.02f}. \n')

```

imagem 4: item #2 do MENU

Em cada item é acessado na memória a tabela das cidades/distância de acordo com a posição na tabela de cada cidade. Cada uma delas tem um índice (utilizando .index()) e é resgatada da memória da ‘list’ “tabela_cidades_distancia” o cruzamento de índices das duas cidades que resulta na distância entre elas.

Na parte final deste bloco 2, é feito o cálculo do custo final em R$ pelo produto do custo por KM e a distância entre as duas cidades do trecho.

Quinto bloco

No quinto bloco é iniciado caso o usuário tenha escolhido o item #3 do MENU. Este bloco tem início semelhante ao quarto bloco (escolha #2). O usuário terá de escolher o custo por KM, não tenha definido anteriormente, antes de iniciar o cálculo de rota (#item 3).

Neste item o usuário deverá escolher uma rota que passe entre 2 ou mais cidades, escrevendo entre vírgulas as respectivas cidades em ordem de origem>destino. Há uma restrição ao usuário caso não utilize uma cidade corretamente, seja por não constar na lista de cidades, ou por erro de digitação, ou caso não utilize as vírgulas conforme recomendado. O programa elimina automaticamente os espaços vazios entre as cidades, portanto, mesmo que se utilize ou não espaço depois das vírgulas, o programa entende e é continuado a execução do quinto bloco.

```
77  if escolha == 3:
78      if custoKM == 0:
79          print('Forneça o valor por KM da viagem no item 1 do MENU.\n')
80          continue
81      rota = []
82      while len(rota) < 2:
83
84          # adicionando as cidades numa list 'rota'.
85          rota = list(input('Digite o nome de duas ou mais cidades, separado por vírgulas e sem acentos: '
86                            ).strip().upper().split(','))
87
88          rota = [c.strip() for c in rota]
89          for cidade in rota:
90              if cidade not in tabela_cidades_distancia[0]:
91                  print(f'A Cidade {cidade} não consta na lista.')
92                  rota = []
93
94      # definindo a posição de cada cidade na tabela conforme posição na list 'tabela_cidades_distancia'
95      for i in range(0, len(rota) - 1):
96          origem_i = rota[i]
97          print(f'Entre {origem_i}', 'e ', end='')
98          destino_i = rota[i + 1]
99          print(f'{destino_i}', end=' ')
100
101         # definindo o índice posição da cidade de origem na tabela das cidades
102         pos_o = tabela_cidades_distancia[0].index(origem_i)
103
104         # definindo o índice posição da cidade de destino na tabela das cidades
105         pos_d = tabela_cidades_distancia[0].index(destino_i)
106
107         # através dos índices da matriz determinar a distância do trecho entre cada cidade na rota
108         distancia_2 = int(tabela_cidades_distancia[pos_o + 1][pos_d])
109         soma3 += distancia_2
110         print(f' a distância entre as cidades é:  {distancia_2} KM.')
```

imagem 5: item #3 do MENU parte 1

Na sequência do bloco cinco (item #3) é somente analisado dois termos (cidades) consecutivas por vez num LOOP do tipo FOR. Este Loop analisa cada trecho da rota uniformemente. E para cada cidade deste trecho é armazenado o índice correspondente na tabela das cidades/distâncias. Consequentemente é armazenada em “distancia_2” a distância entre as cidades no trecho inicial e caso

exista mais trechos nesta rota o valor da distância é acumulada na variável simples “soma3”.

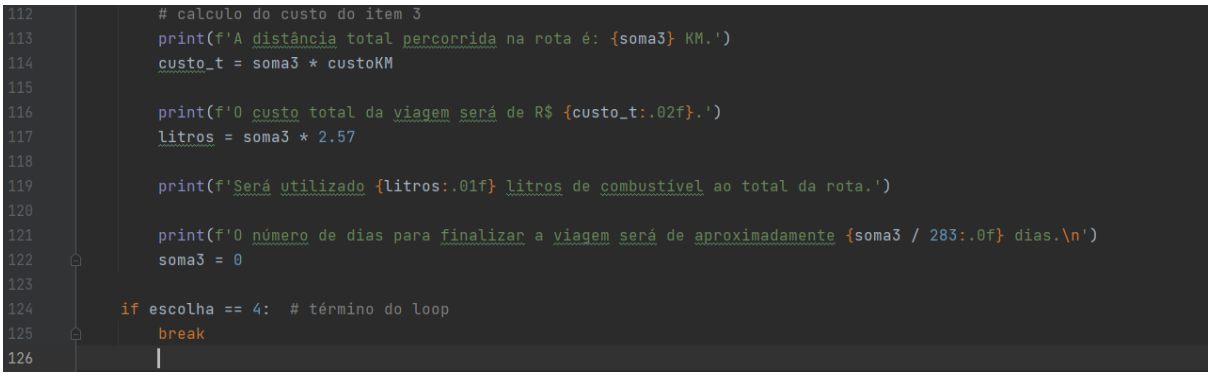

imagem 6: segunda parte do quinto bloco e início do sexto bloco (item #3)

No final do quinto bloco, é mostrado na tela a distância entre cada cidade da rota (fim da imagem 5), a distância total de toda a rota, quantos litros ao total foram gastos. Eu considero o valor de custo do litro por KM algo fora do comum, imagino que inverteram o valor do km/litro para litro/km, mesmo assim, utilizei o valor conforme recomendação no PDF disponibilizado como consulta.
É também feito o cálculo de dias despendidos na viagem calculado pela distância total dividida pela média de km/dia, arredondando para cima.

Sexto bloco

O sexto bloco é utilizado caso o usuário queira sair do programa, para isso, ele deve digitar o valor #4 no MENU inicial. Fazendo esta escolha o LOOP do MENU é encerrado definitivamente.

PRINTS DA EXECUÇÃO DO PROGRAMA

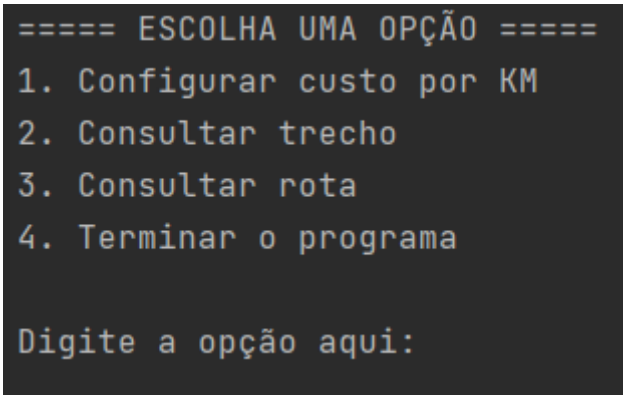

imagem 7: Formato do MENU inicial

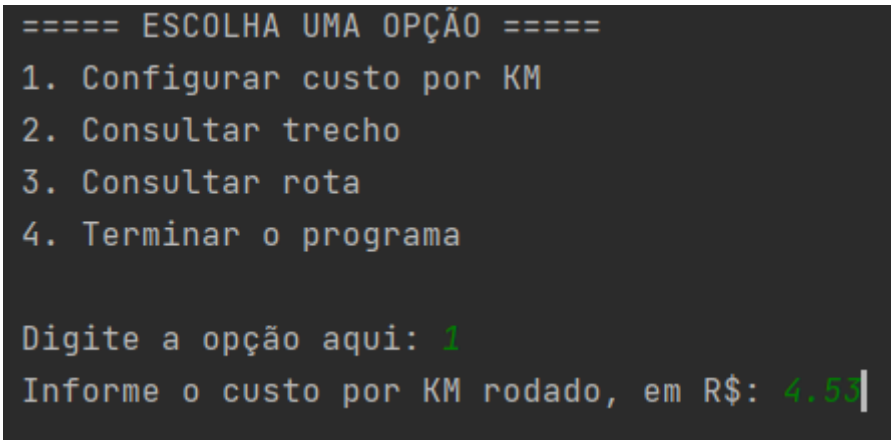

imagem 8: escolhida a opção 1

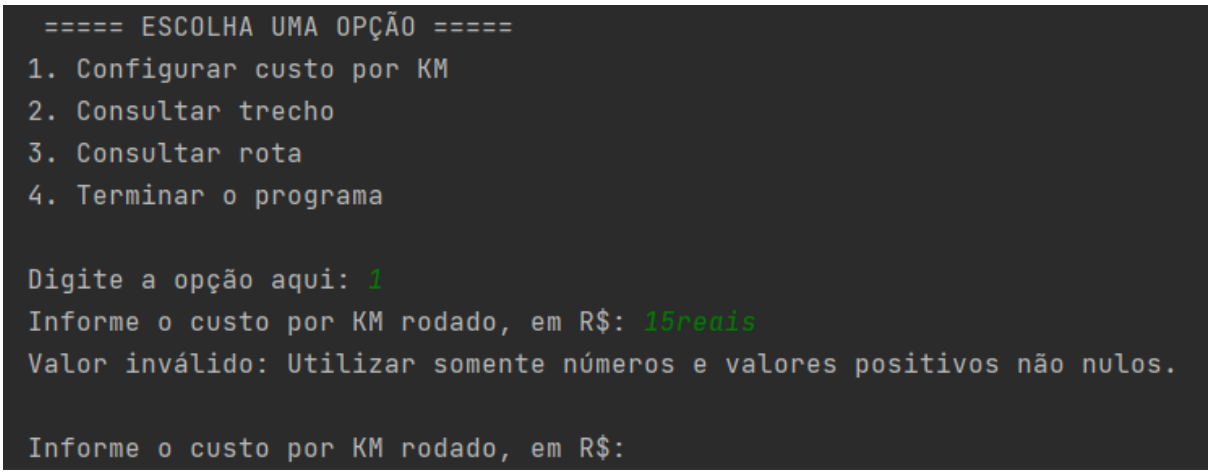

imagem 9: opção 1 escolhida e custo digitado incorretamente

```
 ===== ESCOLHA UMA OPÇÃO =====
1. Configurar custo por KM
2. Consultar trecho
3. Consultar rota
4. Terminar o programa

Digite a opção aqui: 2
Informe a Cidade de Origem: Porto Alegre
Informe a Cidade de Destino: Maceio
O custo da Viagem é R$ 16181.16.
```

imagem 10: opção 2 e dados fornecidos corretamente

```
 ===== ESCOLHA UMA OPÇÃO =====
1. Configurar custo por KM
2. Consultar trecho
3. Consultar rota
4. Terminar o programa

Digite a opção aqui: 2
Informe a Cidade de Origem: coritiba
Informe a Cidade de Destino: pernambuco
Erro de digitação, favor não utilizar acentos ou números.
Informe a cidade de origem novamente: curitiba
Erro de digitação, favor não utilizar acentos ou números.
Informe a cidade de destino novamente: recife
O custo da Viagem é R$ 13943.34.
```

imagem 11: opção 2 com dados fornecidos incorretamente

```
 ===== ESCOLHA UMA OPÇÃO =====
1. Configurar custo por KM
2. Consultar trecho
3. Consultar rota
4. Terminar o programa

Digite a opção aqui: 3
Digite o nome de duas ou mais cidades, separado por vírgulas e sem acentos: porto alegre,florianopolis,curitiba,sao paulo
Entre PORTO ALEGRE e FLORIANOPOLIS  a distância entre as cidades é:  476 KM.
Entre FLORIANOPOLIS e CURITIBA  a distância entre as cidades é:  300 KM.
Entre CURITIBA e SAO PAULO  a distância entre as cidades é:  408 KM.
A distância total percorrida na rota é: 1184 KM.
O custo total da viagem será de R$ 5363.52.
Será utilizado 3042.9 litros de combustível ao total da rota.
O número de dias para finalizar a viagem será de aproximadamente 4 dias.
```

imagem 12: opção 3 com dados fornecidos corretamente

```
 ===== ESCOLHA UMA OPÇÃO =====
1. Configurar custo por KM
2. Consultar trecho
3. Consultar rota
4. Terminar o programa

Digite a opção aqui: 3
Digite o nome de duas ou mais cidades, separado por vírgulas e sem acentos: pernambuco,porto alegre,sao paulo
A Cidade PERNAMBUCO não consta na lista.
Digite o nome de duas ou mais cidades, separado por vírgulas e sem acentos: recife, porto alegre, sao paulo
Entre RECIFE e PORTO ALEGRE  a distância entre as cidades é:  3779 KM.
Entre PORTO ALEGRE e SAO PAULO  a distância entre as cidades é:  1109 KM.
A distância total percorrida na rota é: 4888 KM.
O custo total da viagem será de R$ 22142.64.
Será utilizado 12562.2 litros de combustível ao total da rota.
O número de dias para finalizar a viagem será de aproximadamente 17 dias.
```

imagem 13: opção 3 com dados fornecidos incorretamente

```
 ===== ESCOLHA UMA OPÇÃO =====
1. Configurar custo por KM
2. Consultar trecho
3. Consultar rota
4. Terminar o programa

Digite a opção aqui: 4
Fim do programa!
```

imagem 14: opção 4 término do programa

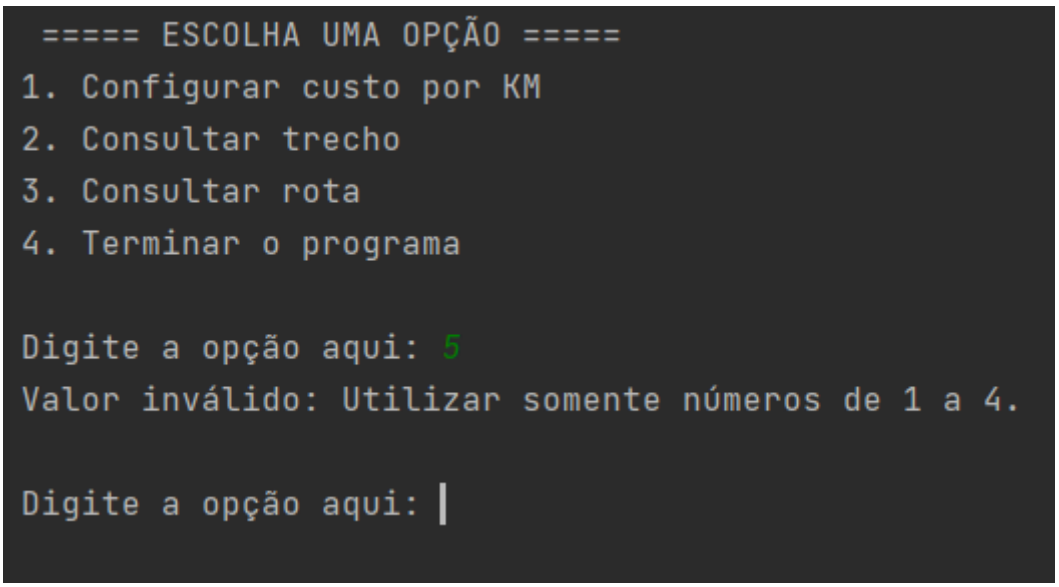

imagem 15: valor inválido para o MENU inicial