ESCOLA POLITÉCNICA

Programa IT Academy – Processo Seletivo – Edição #18

Nome Completo: João Pedro de Moura Medeiros E-mail: joaomoura70718@gmail.com

Etapa 1 – Questões de lógica

Esta seleção possui 15 questões de lógica de caráter eliminatório. As questões são apresentadas no formulário de Exercício Técnico e devem ser respondidas no próprio formulário online, que deverá ser acessado através do link a seguir: https://forms.gle/yZtVcv1b5fCgScLBA

Etapa 2

RESUMO DA SOLUÇÃO

Toda resolução foi feita utilizando: Python (versão 3.8.8), Jupyter Notebook, Visual Studio Code.

A etapa 2 pede para se criar uma simulação para um sistema de transporte, após isso descreve que é necessário ler um arquivo CSV com os dados das distâncias entre as cidades.

A partir dessa ordem, se tem em mente que a leitura do arquivo CSV poderia ser feito com a função da biblioteca Pandas, 'read_csv'. Tendo o CSV lido e plotado na tela, verificasse com alguns métodos da biblioteca Pandas se há dados nulos, os tipos de dados contidos no data frame (DF) e entre outros métodos estatísticos para análise dos dados do DF.

Tendo em vista que o CSV tinha as categorias como cidades, notou-se que os índices poderiam também ser as cidades, já que estavam se correlacionando com as categorias. Logo se foi realizado uma manipulação das colunas, pegando os labels de todas elas e transformando em uma série de dados, para ser inserida no index do DF, assim facilitando muito a manipulação entre os dados durante o progresso do programa.

Continuando a leitura, o enunciado traz informações que uma transportadora tem uma frota composta por caminhões de diferentes modelos com seus respectivos preços e carga. Diante disso foi criado um DF dos caminhões com suas três classes, tipo de caminhão, preço por km e peso de transporte, para facilitar durante a construção das funcionalidades a manipulação dos dados.

Funcionalidades

A funcionalidade 1. [Consultar trechos x modalidade] foi realizada de forma que o usuário deve inserir a cidade de origem e destino dando a ele a distância entre elas, e escolher a modalidade do caminhão, o qual deve um dos números abaixo:
1 - Caminhão de pequeno porte
2 - Caminhão de médio porte
3 - Caminhão de grande porte
Digitando tal número ele localizará, pelo DF criado dos caminhões, o preço por km e seu tipo (pequeno, médio ou grande). Localizando todas a variáveis necessária se fará o cálculo de Preço por Km (R$/km), plotando no final para o usuário toda a informação requerida.

Na funcionalidade 2. [Cadastrar transporte] foi implantado um sistema para que o usuário digite o nome da empresa, os nomes das três cidades, os itens com respectivamente seus pesos e quantidades. Logo após a inserção dos dados, as cidades inseridas pelos usuários vão para uma função a qual retorna a soma das distâncias totais dos trechos das cidades e uma parte da mensagem, como mostrado no enunciado.
Os itens com seus pesos e quantidade são inseridas em outra função, a qual retorna uma lista com as modalidades dos caminhões, para se fazer o cálculo do valor total do transporte dos itens, e uma lista com ‘strings’ para calcular total de veículos deslocados requisitados na funcionalidade 3 do enunciado.
Se fará certas manipulações e cálculos, para aparecer o resto da mensagem, como mostrado no enunciado.
Como a funcionalidade 2 engloba boa parte dos dados solicitados na funcionalidade 3, foi criado um dicionário, o qual receberá os dados solicitados na funcionalidade 3, porém tudo feito na funcionalidade 2, já que todos os métodos podem realizados por ali mesmo e no final armazenando em uma lista global os dicionários com os dados solicitados, fazendo assim que quando se crie um novo dicionário ele vá adicionando na lista global, podendo se utilizar dela na funcionalidade 3.

Por fim na funcionalidade 3. [Dados estatísticos] ele transforma a lista contendo os dicionários criados em um DF, e entrando em uma condição. Se o DF estiver sem dados ele printará 'SEM DADOS ESTATISTÍCOS DOS TRANSPORTES CADASTRADOS! CADASTRE ALGUM TRANSPORTE’ senão irá plotar o DF para o usuário com os dados estatísticos.

Execução do programa

Após toda a informação dada, era hora da criação do programa. Solicitou-se ser feito a criação de um menu com as opções enumeradas nos requisitos, portanto foi feito um bem simples, o qual irá aparece somente no início do programa, quando executado.

As funcionalidades foram feitas cada uma dentro de uma função, criada com o nome ‘def funcionality_X’, sendo X o número da funcionalidade. Após isso elas seriam inseridas dentro de condições, tais condições estariam dentro de um loop o qual seria fator chave para o programa ficar rodando. Caso o usuário digite o número da funcionalidade ela passará pela condição e executará a função, por exemplo:

Usuário digita número 3, PORTANTO SE número_inserido_pelo_usuário == 3 ENTÃO executa funcionalidade 3.

Foi inserido 'Try e Except' em todas as partes a qual o usuário venha inserir determinado dado, caso ele insira de forma incorreta, aparecerá determinada mensagem para ele.

TESTES (aqui você deverá colar capturas de tela de todas as funcionalidades desenvolvidas e realizar comentários, use o espaço que julgar necessário)

FUNCIONALIDADE 1 – Consultar trechos x modalidade

CÓDIGO

Funcionalidade 1. [Consultar trechos x modalidade]

```
''' FUNÇÃO QUE APRESENTA O CÓDIGO DA FUNCIONALIDADE 1 '''

def functionality_1():
    # Inserido um try e um except, caso o usúario insira alguma entrada inválida
    try:
        # Aqui é feito a inserção das cidades de origem e destino
        print("\nINSIRA A CIDADE DE ORIGEM\nExemplo: 'PORTO ALEGRE'\n")
        time.sleep(1)
        # Aqui foi inserido a função input para o usuário inserir a cidade de origem
        city_1 = input('INSIRA A CIDADE DE ORIGEM:').upper()
        print('CIDADE DE ORIGEM INSERIDA')
        print(f'>>>>>{city_1}<<<<<\n')

        # As 5 linhas abaixo farão a mesma coisa, porém será inserida a cidade de destino
        print("INSIRA A CIDADE DE DESTINO\nExemplo: 'SAO PAULO'\n")
        time.sleep(1)
        city_2 = input('INSIRA A CIDADE DE DESTINO:').upper()
        print('CIDADE DE DESTINO INSERIDA!')
        print(f'>>>>>{city_2}<<<<<\n')
```

```python
# Aqui é feito a inserção da modalidade do caminhão
print('-'*60,'\n\nDIGITE O NÚMERO DA MODALIDADE DO CAMINHÃO:\nExemplo: 1 - Caminhão de pequeno porte')
print('          2 - Caminhão de médio porte \n          3 - Caminhão de grande porte\n')
time.sleep(1)
truck_modality = int(input('ESCOLHA A MODALIDADE DO CAMINHÃO:'))
print('NÚMERO DA MODALIDADE DO CAMINHÃO INSERIDO!')
print(f'>>>>>{truck_modality}<<<<<\n')

# Será localizado no 'df_transport' a modalidade requisitadas pelo usurário
list_truck_price = list(df_tranport.iloc[truck_modality-1])
truck = list_truck_price[0]
price = list_truck_price[1]

# Aqui se buscará no 'df' de distância das cidades, os dados das cidades requisitadas pelo usurário
distance_between_cities = df.loc[city_1, city_2]
# Após a pesquisa da distancia das cidades, se multiplicará ela pelo preço, dando assim o Preço por Km (R$/km)
# Foi usado a função 'round' para deixar 2 casas depois da vírgula
price_per_km = round(distance_between_cities*price,2)

# As linhas abaixo mostrarão o output com as cidades, modalidade e calculo conforme
# o exemplo dado na funcionalidade 1
print('-'*60, '\n')
print(f'De {city_1} para {city_2}, utilizando um {truck},')
print(f'a distância é de {distance_between_cities} km e o custo será de R${price_per_km}.\n')
print('-'*60)

time.sleep(1)
```

```python
    except KeyError as c:
        print('-'*80)
        print(f'\nA cidade inserida {c} não está contida no nosso banco de dados\n')
        print('Por favor, insira as cidades sem: \n- Caracteres especiais \n- Acentos \n- Espaços no começo ou fim delas\n')
        print("Exemplo I: insira 'SAO PAULO' ao invés de 'SAO_PAULO'")
        print("Exemplo II: insira 'SAO PAULO' ao invés de 'SÃO PAULO'")
        print("Exemplo III: insira 'SAO PAULO' ao invés de ' SAO PAULO '\n")
        print('Digite novamente 1.[Consultar trechos x modalidade] caso queira consultar a cidade de forma certa\n')
        print('-'*80)
        time.sleep(1)

    except (IndexError, UnboundLocalError):
        print('-'*90)
        print(f'\nO número inserido {truck_modality} da modalidade de caminhão não está contido no nosso banco de dados\n')
        print('Por favor, insira os números (1,2 ou 3), sendo eles sem: \n- Caracteres especiais \n- Acentos \n- Espaços no começo ou fim delas\n')
        print("Exemplo I: insira '2' ao invés de '_2_'")
        print("Exemplo II: insira '2' ao invés de '*2*'\n")
        print('Digite novamente 1.[Consultar trechos x modalidade] caso queira consultar a modalidade de forma certa\n')
        print('-'*90)
        time.sleep(1)
```

EXECUTANDO FUNCIONALIDADE

INSERINDO DADOS DE FORMA CORRETA

```
INSIRA A CIDADE DE ORIGEM
Exemplo: 'PORTO ALEGRE'

CIDADE DE ORIGEM INSERIDA
>>>>>PORTO ALEGRE<<<<<
```

```
INSIRA A CIDADE DE DESTINO
Exemplo: 'SAO PAULO'

CIDADE DE DESTINO INSERIDA!
>>>>>SAO PAULO<<<<<
```

```
DIGITE O NÚMERO DA MODALIDADE DO CAMINHÃO:
Exemplo: 1 - Caminhão de pequeno porte
         2 - Caminhão de médio porte
         3 - Caminhão de grande porte

NÚMERO DA MODALIDADE DO CAMINHÃO INSERIDO!
>>>>>1<<<<<
```

RESULTADO

```
De PORTO ALEGRE para SAO PAULO, utilizando um Caminhão de porte PEQUENO,
a distância é de 1109 km e o custo será de R$5400.83.
```

INSERINDO A CIDADE DE FORMA INCORRETA

```
INSIRA A CIDADE DE ORIGEM
Exemplo: 'PORTO ALEGRE'

CIDADE DE ORIGEM INSERIDA
>>>>>POA<<<<<

INSIRA A CIDADE DE DESTINO
Exemplo: 'SAO PAULO'

CIDADE DE DESTINO INSERIDA!
>>>>>SAO PAULO<<<<<
```

```
DIGITE O NÚMERO DA MODALIDADE DO CAMINHÃO:
Exemplo: 1 - Caminhão de pequeno porte
         2 - Caminhão de médio porte
         3 - Caminhão de grande porte

NÚMERO DA MODALIDADE DO CAMINHÃO INSERIDO!
>>>>>3<<<<<

--------------------------------------------------------------------------------

A cidade inserida 'POA' não está contida no nosso banco de dados

Por favor, insira as cidades sem:
- Caracteres especiais
- Acentos
- Espaços no começo ou fim delas

Exemplo I: insira 'SAO PAULO' ao invés de 'SAO_PAULO'
Exemplo II: insira 'SAO PAULO' ao invés de 'SÃO PAULO'
Exemplo III: insira 'SAO PAULO' ao invés de '  SAO PAULO  '

Digite novamente 1.[Consultar trechos x modalidade] caso queira consultar a cidade de forma certa
```

INSERINDO A MODALIDADE DE FORMA INCORRETA

```
INSIRA A CIDADE DE ORIGEM
Exemplo: 'PORTO ALEGRE'

CIDADE DE ORIGEM INSERIDA
>>>>>PORTO ALEGRE<<<<<

INSIRA A CIDADE DE DESTINO
Exemplo: 'SAO PAULO'

CIDADE DE DESTINO INSERIDA!
>>>>>CURITIBA<<<<<
```

```
DIGITE O NÚMERO DA MODALIDADE DO CAMINHÃO:
Exemplo: 1 - Caminhão de pequeno porte
         2 - Caminhão de médio porte
         3 - Caminhão de grande porte

NÚMERO DA MODALIDADE DO CAMINHÃO INSERIDO!
>>>>>4<<<<<

--------------------------------------------------------------------------------

O número inserido 4 da modalidade de caminhão não está contido no nosso banco de dados

Por favor, insira os números (1,2 ou 3), sendo eles sem:
- Caracteres especiais
- Acentos
- Espaços no começo ou fim delas

Exemplo I: insira '2' ao invés de '_2_'
Exemplo II: insira '2' ao invés de '*2*'

Digite novamente 1.[Consultar trechos x modalidade] caso queira consultar a modalidade de forma certa
```

FUNCIONALIDADE 2 – Cadastrar transporte

CÓDIGO

```
''' NESSA CELULA TERÁ DUAS FUNÇÕES, AS QUAIS FAZEM PARTE DA FUNÇÃO FUNCIONALIDADE 2 '''

# Função que identifica a distância entre as cidades e retorna o valor total delas
def model_identify(city_1, city_2, city_3):
    # df.loc[index, coluna] irá procurar no df o dados correspondente ao index e coluna
    distance_between_cities_1 = df.loc[city_1, city_2]
    distance_between_cities_2 = df.loc[city_2, city_3]
    sum_distance = distance_between_cities_1 + distance_between_cities_2

    print(f'de {city_1} para {city_2}, a distância a ser percorrida é de {distance_between_cities_1} km')
    print(f'e de {city_2} para {city_3}, a distância a ser percorrida é de {distance_between_cities_2} km')
    print(f'somando um total de {sum_distance} km, para transporte dos produtos')

    return sum_distance
```

```
# Função para calcular o peso dos itens pelas suas quantidades e a partir disso escolhendo os caminhões para transporte
def cost_transport(weight_item_list, quantity_item_list, items_list):
    list_total_weight_item = list(map(lambda x, y: (x*y), weight_item_list, quantity_item_list))
    total_weight = sum(list_total_weight_item)

    # Modalidades dos caminhões
    truck_modality_1 = df_tranport['Itens'][1]
    truck_modality_2 = df_tranport['Itens'][2]
    truck_modality_3 = df_tranport['Itens'][3]
    list_modality = [truck_modality_1, truck_modality_2, truck_modality_3]
    # Peso de transporte dos caminhões
    truck_weight_1 = df_tranport['Peso de transporte(kg)'][1]
    truck_weight_2 = df_tranport['Peso de transporte(kg)'][2]
    truck_weight_3 = df_tranport['Peso de transporte(kg)'][3]
```

```
    # Aqui será feito a contagem de quantos caminhões serão necessários para o transporte da carga se baseando no peso dela
    list_transport_truck = []
    while total_weight > 0:
        if total_weight >= truck_weight_3:
            total_weight = total_weight - truck_weight_3
            list_transport_truck.append(truck_modality_3)
        elif total_weight >= truck_weight_2:
            total_weight = total_weight - truck_weight_2
            list_transport_truck.append(truck_modality_2)
        else:
            total_weight = total_weight - truck_weight_1
            list_transport_truck.append(truck_modality_1)

    truck_count_1 = list_transport_truck.count(truck_modality_1)
    truck_count_2 = list_transport_truck.count(truck_modality_2)
    truck_count_3 = list_transport_truck.count(truck_modality_3)
```

```
    # Criado um dicionário com a modalidade do caminhão como chave e o quantidade que necessitará para o transporte
    dict_truck_count = ({truck_modality_1: truck_count_1, truck_modality_2: truck_count_2, truck_modality_3: truck_count_3})
    values = list(dict_truck_count.values())

    list_truck_transport_count = list(map(lambda x, y: f'{x} {y}' if x != 0 else None , values, list_modality ))
    list_truck_transport_count = list(filter(lambda x: x is not None, list_truck_transport_count))

    ####print(list_truck_transport_count)
    str_truck_tranport_count = ', '.join(list_truck_transport_count)
    str_items_list = ', '.join(items_list)

    ####print(str_truck_tranport_count)
    ####print(list_transport_truck, '\n')
    print(f'{str_items_list} será necessário utilizar')
    print(f'{str_truck_tranport_count},\nde forma a resultar no menor custo de transporte por km rodado.')

    return (list_transport_truck, list_truck_transport_count)
```

```python
''' FUNÇÃO QUE APRESENTA O CÓDIGO DA FUNCIONALIDADE 2 '''

# Lista global criada para adicionar os dados estatísticos do cadastro de transporte
list_transport_register = []

def functionality_2():
    # Inserido um try e um except, caso o usúario insira alguma entrada inválida
    try:
        print('\nFAÇA O CADASTRO DA SUA EMPRESA! INSIRA O NOME DE SUA EMPRESA\n')
        print('-'*80)
        # Input para o usuário cadastrar sua empresa
        company = str(input('\nFAÇA O CADASTRO DA SUA EMPRESA! INSIRA O NOME DE SUA EMPRESA'))
        print('\nNOME DA EMPRESA INSERIDO!')
        print(f'>>>>>{company}<<<<<\n')
        print('-'*80)
        time.sleep(2)

        # Usuário irá inserir as cidades desejadas, tendo ela sendo adicionadas em uma lista
        city_list = []
        print('\nINSIRA AS CIDADES DESEJADAS!\n')
        for i in range(1,4):
            city_isf = ['INICIAL', 'DE PARADA', 'FINAL']
            value_city = city_isf[i-1]
            print(f'INSIRA A CIDADE {value_city}')
            city_list.append(str(input(f'INSIRA A CIDADE {value_city}').upper()))
            city_inserted = city_list[i-1]
            print(f'CIDADE {value_city} INSERIDA FOI: {city_inserted}\n')
        print('-'*80,'\n')
        time.sleep(2)
```

```python
        # Usuário irá inserir os itens desejadas com respectivamente seus pesos e quantidades
        # tendo cada um deles adicionados em uma lista
        items_list = []
        weight_item_list = []
        quantity_item_list = []
        print('INSIRA OS NOMES DOS ITENS DESEJADOS COM SEUS PESOS E QUANTIDADES!\n')
        time.sleep(1)
```

```python
        # Dentro desse for será requsitado colocar o nome do item, seu peso e quantidade
        for i in range(1,5):
            print(f'\nINSIRA O NOME DO ITEM NÚMERO {i}')
            items_list.append(str(input(f'INSIRA O NOME DO ITEM NÚMERO {i}')))
            item_inserted = items_list[i-1]
            time.sleep(1)
            try:
                print(f'INSIRA O PESO DO ITEM NÚMERO {i}')
                weight_item_input = str(input(f'INSIRA O PESO DO ITEM NÚMERO {i}'))
                weight_item_list.append(float(weight_item_input.replace(",", ".")))
                weight_item_inserted = weight_item_list[i-1]
                time.sleep(1)
                print(f'INSIRA A QUANTIDADE DO ITEM NÚMERO {i}')
                quantity_item_input = int(input(f'INSIRA A QUANTIDADE DO ITEM NÚMERO {i}'))
                quantity_item_list.append(quantity_item_input)
                quantity_item_inserted = quantity_item_list[i-1]
                time.sleep(1)
                print(f'\nO ITEM NÚMERO {i} INSERIDO FOI >>>{item_inserted}<<< E SEU PESO É DE >>>{weight_item_inserted}kg<<<')
                print(f'COM >>>{quantity_item_inserted}<<< DE QUANTIDADE\n')
            except (ValueError,KeyError):
                print()
                print('-'*90)
                print('\nNUMERO INSERIDO DE FORMA ERRADA!\n')
                print('PARA A INSERÇÃO DE PESO, É APENAS POSSÍVEL NÚMEROS!')
                print('PARA A INSERÇÃO DE QUANTIDADE, É APENAS POSSÍVEL NÚMEROS INTEIROS(SEM PONTO OU VÍRGULA)!\n')
                print('-'*90)
                time.sleep(1)
            time.sleep(1)
        print('-'*80,'\n')
```

```
            sum_distance = int(model_identify(city_list[0], city_list[1], city_list[2]))
            result = cost_transport(weight_item_list, quantity_item_list, items_list)

            # Interando sobre a lista retornada da função cost_transport, pegará as modalidade dos caminhões tais como
            # ['Caminhão de porte GRANDE', 'Caminhão de porte MÉDIO', 'Caminhão de porte PEQUENO']
            list_total_prices_per_way = []
            for i in result[0]:
                price_per_km = float(df_tranport.loc[df_tranport['Itens'] == i]['Preco por Km(R$/km)'])
                total_price = round(price_per_km*sum_distance, 2)
                list_total_prices_per_way.append(total_price)

            # Aqui será feito o cálculo dos valores totais do transporte e os valores unitários
            var_total_price = round(sum(list_total_prices_per_way),2)
            sum_quantity_item_list= sum(quantity_item_list)
            var_unitary_price = round(var_total_price/sum_quantity_item_list,2)
            print(f'O valor total do transporte dos itens é R$ {var_total_price},')
            print(f'sendo R$ {var_unitary_price} é o custo unitário médio.\n')
            print('-'*80,'\n')

            # Aqui será feito o calculo do 'Custo médio por tipo de produto'
            set_items = list(set(items_list))
            mean_cust_per_product = round(var_total_price/len(set_items),2)
```

```
            # Aqui será feito o calculo do 'Custo total para cada modalidade de transporte (R$)'
            # Se retira valores duplicados
            set_modality = list(set(result[0]))
            # Ordena em forma crescente os valores da list_total_prices_per_way
            list_total_prices_per_way.sort()
            set_sort_total_prices = list(set(list_total_prices_per_way))
            # Junta os valores pelo index de cada lista, isso retornará o 'Custo total para cada modalidade de transporte (R$)'
            cust_per_modality = tuple(zip(set_modality, set_sort_total_prices))

            # Aqui será feito o método para pegar o total de veículos deslocados
            list_nums = []
            for i in result[1]:
                # É utilizado regex para filtrar o número da string
                num = (re.findall('[0-9]+', i))
                list_nums.append(num[-1])
            # Se faz a soma de todos o números, dando assim o 'Número total de veículos deslocados'
            total_trucks = 0
            for valor in list_nums:
                total_trucks += int(valor)
```

```
            # Aqui será feito o armazenamento dos dados cadastrados em dicionário
            # Para que possa exibir um relatório dos transportes até então cadastrados na funcionalidade 3
            dict_transport = dict({'Empresa': company,
                                   'Custo total (R$)': var_total_price,
                                   # Custo médio por trecho ??? Não foi possível entender do enunciado, logo se fará a média por cada trecho
                                   'Custo por trecho (R$)': round(var_total_price/3,2),
                                   'Custo médio por km (R$)': round(var_total_price/sum_distance,2),
                                   'Custo médio por tipo de produto (R$)': mean_cust_per_product,
                                   'Custo total por trecho (R$)': var_total_price,
                                   'Custo total para cada modalidade de transporte (R$)': cust_per_modality,
                                   'Número total de veículos deslocados': total_trucks,
                                   'Total de itens Transportados': sum(quantity_item_list)})

            # Adicionará o dicionário em uma lista global, para guardar na memória, podendo utilizar na funcionalidade 3
            list_transport_register.append(dict_transport)
            time.sleep(2)
```

```

        except KeyError as c:
            print(f'A cidade inserida {c} não está contida no nosso banco de dados\n')
            print('Por favor, insira as cidades sem: \n- Caracteres especiais \n- Acentos \n- Espaços no começo ou fim delas\n')
            print("Exemplo I: insira 'SAO PAULO' ao invés de 'SAO_PAULO'")
            print("Exemplo II: insira 'SAO PAULO' ao invés de 'SÃO PAULO'")
            print("Exemplo III: insira 'SAO PAULO' ao invés de ' SAO PAULO '\n")
            print('Digite novamente 2.[Cadastrar transporte] caso queira consultar a cidade de forma certa\n')
            print('-'*80)
            time.sleep(1)

```

EXECUTANDO FUNCIONALIDADE

INSERINDO DADOS DE FORMA CORRETA

```
FAÇA O CADASTRO DA SUA EMPRESA! INSIRA O NOME DE SUA EMPRESA

--------------------------------------------------------------------------------

NOME DA EMPRESA INSERIDO!
>>>>>TikStop<<<<<

--------------------------------------------------------------------------------
```

```
INSIRA AS CIDADES DESEJADAS!

INSIRA A CIDADE INICIAL
CIDADE INICIAL INSERIDA FOI: PORTO ALEGRE

INSIRA A CIDADE DE PARADA
CIDADE DE PARADA INSERIDA FOI: FLORIANOPOLIS

INSIRA A CIDADE FINAL
CIDADE FINAL INSERIDA FOI: CURITIBA
```

```
INSIRA OS NOMES DOS ITENS DESEJADOS COM SEUS PESOS E QUANTIDADES!


INSIRA O NOME DO ITEM NÚMERO 1
INSIRA O PESO DO ITEM NÚMERO 1
INSIRA A QUANTIDADE DO ITEM NÚMERO 1

O ITEM NÚMERO 1 INSERIDO FOI >>>celulares<<< E SEU PESO É DE >>>0.5kg<<<
COM >>>300<<< DE QUANTIDADE


INSIRA O NOME DO ITEM NÚMERO 2
INSIRA O PESO DO ITEM NÚMERO 2
INSIRA A QUANTIDADE DO ITEM NÚMERO 2

O ITEM NÚMERO 2 INSERIDO FOI >>>geladeiras<<< E SEU PESO É DE >>>20.0kg<<<
COM >>>50<<< DE QUANTIDADE
```

```
INSIRA O NOME DO ITEM NÚMERO 3
INSIRA O PESO DO ITEM NÚMERO 3
INSIRA A QUANTIDADE DO ITEM NÚMERO 3

O ITEM NÚMERO 3 INSERIDO FOI >>>freezers<<< E SEU PESO É DE >>>10.5kg<<<
COM >>>70<<< DE QUANTIDADE


INSIRA O NOME DO ITEM NÚMERO 4
INSIRA O PESO DO ITEM NÚMERO 4
INSIRA A QUANTIDADE DO ITEM NÚMERO 4

O ITEM NÚMERO 4 INSERIDO FOI >>>luminárias<<< E SEU PESO É DE >>>10.0kg<<<
COM >>>2000<<< DE QUANTIDADE
```

RESULTADO

```
de PORTO ALEGRE para FLORIANOPOLIS, a distância a ser percorrida é de 476 km
e de FLORIANOPOLIS para CURITIBA, a distância a ser percorrida é de 300 km
somando um total de 776 km, para transporte dos produtos
celulares, geladeiras, freezers, luminárias será necessário utilizar
2 Caminhão de porte PEQUENO, 2 Caminhão de porte GRANDE,
de forma a resultar no menor custo de transporte por km rodado.
O valor total do transporte dos itens é R$ 50145.12,
sendo R$ 20.72 é o custo unitário médio.
```

INSERINDO A CIDADE DE FORMA INCORRETA

```
NOME DA EMPRESA INSERIDO!
>>>>>DELL<<<<<

--------------------------------------------------------------------------------

INSIRA AS CIDADES DESEJADAS!

INSIRA A CIDADE INICIAL
CIDADE INICIAL INSERIDA FOI: POA

INSIRA A CIDADE DE PARADA
CIDADE DE PARADA INSERIDA FOI: CURITIBA

INSIRA A CIDADE FINAL
CIDADE FINAL INSERIDA FOI: SAO PAULO
```

```
A cidade inserida 'POA' não está contida no nosso banco de dados

Por favor, insira as cidades sem:
- Caracteres especiais
- Acentos
- Espaços no começo ou fim delas

Exemplo I: insira 'SAO PAULO' ao invés de 'SAO_PAULO'
Exemplo II: insira 'SAO PAULO' ao invés de 'SÃO PAULO'
Exemplo III: insira 'SAO PAULO' ao invés de '  SAO PAULO  '

Digite novamente 2.[Cadastrar transporte] caso queira consultar a cidade de forma certa
```

INSERINDO O PESO E QUANTIDADE DE FORMA INCORRETA

```
CELULAR
INSIRA O PESO DO ITEM
```

```
NUMERO INSERIDO DE FORMA ERRADA!

PARA A INSERÇÃO DE PESO, É APENAS POSSÍVEL NÚMEROS!
PARA A INSERÇÃO DE QUANTIDADE, É APENAS POSSÍVEL NÚMEROS INTEIROS(SEM PONTO OU VÍRGULA)!
```

```
DOIS
INSIRA A QUANTIDADE DO ITEM
```

```
NUMERO INSERIDO DE FORMA ERRADA!

PARA A INSERÇÃO DE PESO, É APENAS POSSÍVEL NÚMEROS!
PARA A INSERÇÃO DE QUANTIDADE, É APENAS POSSÍVEL NÚMEROS INTEIROS(SEM PONTO OU VÍRGULA)!
```

FUNCIONALIDADE 3 – Dados estatísticos

CÓDIGO

Funcionalidade 3.[Dados estatísticos]

```python
''' FUNÇÃO QUE APRESENTA O CÓDIGO DA FUNCIONALIDADE 3 '''

def functionality_3():
    # Pega a lista global e a transforma em um df para a visualização dos dados estatísticos
    df_register = pd.DataFrame(list_transport_register)
    # Caso o df esteja vazio printará a seguinte frase
    if df_register.empty:
        print()
        print('SEM DADOS ESTATISTÍCOS DOS TRANSPORTES CADASTRADOS! CADASTRE ALGUM TRANSPORTE\n')
    # Senão mostra na tela para o usuário os dados estatisticos
    else:
        print()
        print(df_register.to_string(),'\n')
    print('-'*80)

# Retire a # abaixo caso queirar testar a funcionalidade
functionality_3()
```

✓ 0.0s

EXECUTANDO FUNCIONALIDADE

PLOTAGEM DOS DADOS CADASTRADOS

Empresa TikStop

```
   Empresa  Custo total (R$)  Custo por trecho (R$)  Custo médio por km (R$)
0  TikStop          50145.12                16715.04                   64.62
```

```
Custo médio por tipo de produto (R$)  Custo total por trecho (R$)
                             12536.28                     50145.12
```

```
                                Custo total para cada modalidade de transporte (R$)
((Caminhão de porte PEQUENO, 3779.12), (Caminhão de porte GRANDE, 21293.44))
```

```
Número total de veículos deslocados  Total de itens Transportados
                                  4                           2420
```

Empresa LeMour

	Empresa	Custo total (R$)	Custo por trecho (R$)	Custo médio por km (R$)
0	TikStop	50145.12	16715.04	64.62
1	LeMour	178344.04	59448.01	58.84

Custo médio por tipo de produto (R$)	Custo total por trecho (R$)
12536.28	50145.12
44586.01	178344.04

Custo total para cada modalidade de transporte (R$)

((Caminhão de porte PEQUENO, 3779.12), (Caminhão de porte GRANDE, 21293.44))

((Caminhão de porte PEQUENO, 14760.97), (Caminhão de porte GRANDE, 36129.52), (Caminhão de porte MÉDIO, 83170.64))

Número total de veículos deslocados	Total de itens Transportados
4	2420
6	898

PROGRAMA - CÉLULA QUE IRÁ RODAR TODAS AS FUNCIONALIDADES

```
''' AQUI SERÁ O CÓDIGO QUE IRÁ EXECUTAR TODO O PROGRAMA, JUNTANDO TODOS OS CÓDIGOS EM UM SÓ '''

# Chamando a função menu
menu()

list_functionality = ['1.[Consultar trechos x modalidade]',
                      '2.[Cadastrar transporte]',
                      '3.[Dados estatísticos]',
                      '4.[Finalizar o programa]']

i = int
# Criado um while para ficar rodando o programa
while i != 4:
```

```
    # Inserido um try e um except, caso o usúario insira alguma entrada inválida
    try:
        print("\nDIGITE O NÚMERO DA FUNCIONALIDADE\n")
        # Usuário deve digitar o número de alguma funcionalidade
        i = int(input("DIGITE O NÚMERO DA FUNCIONALIDADE:"))
        number_functionality = list_functionality[i-1]
        print(f"O NÚMERO DA FUNCIONALIDADE ESCOLHIDO FOI\n>>> {number_functionality}\n")
        time.sleep(2)
        print('-'*60)
        if (i == 1):
            # Executará o código da funcionalidade
            functionality_1()
        elif (i == 2):
            # Executará o código da funcionalidade
            functionality_2()
        elif (i == 3):
            # Executará o código da funcionalidade
            functionality_3()
        elif (i == 4):
            print("\nO PROGRAMA FOI FINALIZADO! ATÉ LOGO!")
            break
```

```
        else:
            print("\nERRO DE NUMERAÇÃO, DIGITE APENAS OS NÚMEROS DAS FUNCIONALIDADES (1, 2, 3, 4)\n")
            print('-'*60)
            time.sleep(1)
    except (ValueError, IndexError):
        print('-'*60)
        print("\nDIGITE APENAS NÚMEROS (1, 2, 3 OU 4)\n")
        print('-'*60)
        time.sleep(1)
```

OBS: em certa parte durante a execução do programa, devido ao número limite de linhas do output do jupyter, ficará sem plotar nada e aparecerá essa mensagem:

```
Output exceeds the size limit. Open the full output data in a text editor
```

Clique em 'in a text editor' para abrir em uma nova aba e poder executar normalmente o código

Para executar o programa é necessário rodar todas as células do jupyter notebook

AUTOAVALIAÇÃO

Você concluiu a implementação de 100% das funcionalidades solicitadas?

(X) Sim ( ) Não

Para as 3 principais funcionalidades solicitadas, como você avalia a sua solução?

Marque um 'X'.

	Inexistente/ Insuficiente	Pouco satisfeito(a)	Satisfeito(a)	Muito satisfeito(a)
Funcionalidade 1				X
Funcionalidade 2			X	
Funcionalidade 3				X

Principais dificuldades

Não apresentei muitas dificuldades, tendo em vista que evolui muito em questão de lógica e programação desde que fiz os últimos testes técnicos do IT Academy. Pode se dizer que a maior dificuldade não foi em algum assunto o qual não dominava, pelo contrário, sabia todas as ferramentas as quais utilizar para resolver as questões, a dificuldade em si estava na interpretação do enunciado.
Sinto que em certas partes do enunciado faltavam alguma informação necessária, tal como na parte do enunciado que explica a funcionalidade 3. [dados estatísticos] - "Para cada um deverá ser apresentado o custo total, **o custo por trecho**, o custo médio por km, o custo médio por tipo de produto, **o custo total por trecho**,". 'O custo por trecho' não é igual a 'o custo total por trecho'?
Enfim, acredito que eu tenha evoluído muito, percebendo que fiz de forma muito eficiente sem quebrar tanto a cabeça.

Desempenho Geral

Após de ter finalizado a construção do programa, acredito que meu desempenho tenha sido muito bom, consegui executar todas as funcionalidades de forma eficiente, tive uma organização boa em relação as tarefas. Entretanto acredito que meu código ficou com uma complexidade alta, porém não tive muito mais tempo para aperfeiçoar ele.
A biblioteca do Python, Pandas, me proporcionou muita facilidade na programação, tanto na leitura, análise, processamento e manipulação dos dados.
Utilizei outras bibliotecas tais como 'time' e 'regex', sendo necessária para o meu código. 'Time' foi útil para a parte de plotagem dos textos do menu, já 'regex' utilizada em apenas uma linha, para o tratamento de certa 'string', sinto que poderia ter usado bem mais ao decorrer do código, mas também não queria abusar tanto da ferramenta, decidindo usar as ferramentas nativas do Python e uma lógica simples.

Obrigado por participar deste processo seletivo.
Salve o documento em PDF com o seu nome completo.