RELATÓRIO DELL IT ACADEMY

João Vitor Boer Abitante

PUCRS

Introdução:

O arquivo enviado no e-mail para resolução pode ser resumido assim: Temos um CSV com dados de bolsas e a partir dele devemos resolver algumas perguntas propostas na descrição, tais como:

1. Consultar a bolsa zero do ano solicitado.
2. Codificar o nome e retornar informações sobre a pessoa solicitada.
3. Consultar a média anual do ano inserido.
4. Rankear as três maiores e menores bolsas.
5. Permitir que o usuário finalize o programa.

Para a resolução dos problemas propostos utilizamos como ambiente de programação o Jupyter Notebook e o VSCode, com o auxilio da biblioteca *Pandas* para manipulação de dados, *Unidecode* para tratamento de acentuação em palavras e *Os* para encerrar o programa.

Organizando o Ambiente:

Primeiramente, escrevemos um check-list com os passos que devemos seguir.
Exemplo:

1. Ler e entender o problema.
2. Criar uma pasta para organizar os arquivos.

3. Criar um arquivo principal e um de teste.

4. Ler e importar a base de dados.

5. Analisar os dados.

6. Tratar os dados.

7. Criar menu de interação com o usuário.

Felizmente, não foi-se necessário fazer nenhum tipo de tratamento, pois utilizando comando .info(), percebe-se que não há non-nulls´s (dados faltantes) e os tipos dos dados estão corretos.

```
df.info()

RangeIndex: 80015 entries, 0 to 80014
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   NM_BOLSISTA              80015 non-null  object
 1   CPF_BOLSISTA             80015 non-null  object
 2   NM_ENTIDADE_ENSINO       80015 non-null  object
 3   ME_REFERENCIA            80015 non-null  int64
 4   AN_REFERENCIA            80015 non-null  int64
 5   SG_DIRETORIA             80015 non-null  object
 6   SG_SISTEMA_ORIGEM        80015 non-null  object
 7   CD_MODALIDADE_SGB        80015 non-null  int64
 8   DS_MODALIDADE_PAGAMENTO  80015 non-null  object
 9   CD_MOEDA                 80015 non-null  object
 10  VL_BOLSISTA_PAGAMENTO    80015 non-null  int64
dtypes: int64(4), object(7)
```

Além disso, um check-list para cada uma das instruções foi realizado. Exemplo considerando a pergunta 1:

1. Pegar o ano e verificar se faz parte dos limites da tabela.

2. Se for verdade, localizar todas as linhas iguais ao ano informado.

3. Caso contrário, tratar valor inválido.

4. Exibir as informções corretas

1. Consultar a bolsa zero do ano solicitado:

Para encontrar o respectivo bolsista, primeiramente perguntamos o ano desejado, e então o colocamos dentro de uma verificação , para ver se ele está entre os anos disponiveis na tabela. Se for verdade, o programa localiza e armazena em um novo Data Frame todas as linhas em que o ano é igual ao ano informado, através do comando .loc(). Caso contrário, o usuário receberá uma mensagem de erro.

Então, como a orientação é exibir somente os dados do bolsista zero, na hora de printar usa-se do print por indexação, aonde o comando iloc[linha, coluna] é utilizado. Como nota-se no código, o valor que vai na linha é sempre 0, dado o fato de queremos o primeiro bolsista daquele ano. E o valor que vai na coluna depende da informação que estamos procurando. Ex: para pegar o valor da bolsa, devemos utilizar do valor -1, pois a coluna VL_BOSISTA_PAGAMENTO é a ultima. Logo, o valor a ser impresso será aquele que está na linha zero na coluna "-1" (VL_BOSISTA_PAGAMENTO).

Resultado:

```
Informe o ano para consultar a primeira bolsa: 2016

 O bolsista 0 em 2016:

                    Nome = ALEXANDRE RIBEIRO NETO
                    CPF  = ***.195.647-**
                    Nome da Entidade de Ensino = UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
                    Valor da Bolsa = 765
```

Testes:

Primeiramente os testes foram realizados com entradas de dados incorretos, feitos a mão. Logo após, em um arquivo separado, foi realizado o teste unitário. Nesse teste, o programa compara todos os nomes obtidos como resultado com seus respectivos anos que foram solicitados. Caso todos estejam corretos, o programa retorna True.

```
anos = [2014,2015,2013,2016]
achou = False

for ano in anos:
    for i, info in enumerate(df['AN_REFERENCIA']):
        if ano == info:
            index = i
            if df.iloc[index, 0] == 'JOSE QUERGINALDO BEZERRA': achou = True #2013
            if df.iloc[index, 0] == 'CAROLINE ALCANTARA DUARTE': achou = True #2014
            if df.iloc[index, 0] == 'CICERO SARAIVA SOBRINHO': achou = True #2015
            if df.iloc[index, 0] == 'ALEXANDRE RIBEIRO NETO': achou = True #2016


print(achou)
```

```
True
```

Como os anos no CSV estavam todos em ordem, realizei uma também consulta manual para verificar se os resultados batiam.

2. Codificar o nome e retornar informações:

Sem dúvidas, essa foi a operação mais desafiadora. Diversos problemas foram enfrentados e superados durante o desenvolvimento. Também confesso que me empolguei e não segui meu check-list, o que me levou a fazer diversar confusões e ter de refazer novamente a lógica.

Para começar, o usuário informa o nome do jeito que desejar. E após isso, cada palavra da string informada é separa e colocada em um vetor. Então é chamado a função editar_texto() que trata e retorna cada item desse vetor, removendo acentos e carateres especiais.

Visto que o nome informado pode ser somente parte de um nome correspondente na tabela, temos então que achar seu correspondente. Para isso, o programa verifica cada nome no Data Frame e compara com o nome informado já tratado. Se alguém na tabela conter todos os nomes que foram informados pelo usuário, isso significa que achamos nossos bolsistas e agora podemos codificar seus nomes completos.

Para codificar, o programa chama a função codificador() que executa a codificação em 4 passos:

1. Faz a troca da primeira com a ultima letra.
2. Inverte a string se ela tiver mais de 3 letras.
3. Faz a troca das letras, A => B.
4. Retorna a string codificada.

Nessa hora, eu percebi que estava com um erro, que consegui arrumar mais tarde, exemplo: se o nome informado foi Maria da Silva, e tiver na tabela uma pessoa chamada Maria da Silva Nunes e outra pessoa chamada Maria da Silva Nunes Porto, o programa vai considerar as duas opções como válida, porém o meu programa estava considerando só a primeira pessoa como válida. Então a solução foi pegar todos os nomes encontrados, codificá-los e armazena-los em outro vetor separado, e então excluir os nomes não-codificados duplicados. Assim, fui capaz de printar todas as informações corretamente.

Após isso, o programa localiza no Data Frame os dados pedidos no enunciado respectivos ao bolsista e armazena em um vetor que posteriormente é concatenado e vira um Data Frame. Então o programa exibe eles na tela os dados dos bolsistas. Obs: o programa printa o mesmo bolsista porém em anos diferentes.

Resultado:

```
Nome Codificado: KOBJMVB GBUJFST MFQPT
Ano: 2014
Entidade de Ensino: UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Valor da Bolsa: 765



Nome Codificado: KOBJMVB GBUJFST MFQPT
Ano: 2013
Entidade de Ensino: UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Valor da Bolsa: 765
```

```
Nome Codificado: KOBJMVB OJBWPT MFQPT
Ano: 2013
Entidade de Ensino: UNIVERSIDADE FEDERAL DO ESPIRITO SANTO
Valor da Bolsa: 765



Nome Codificado: KOBJMVB MFQPT GSFWBP
Ano: 2013
Entidade de Ensino: UNIVERSIDADE DO ESTADO DA BAHIA
Valor da Bolsa: 765
```

Testes:

Para começar, entrei com dados inexistentes para ver se o programa informava a mensagem correta. Após isso, em um arquivo separado, criei uma função que manda todos os nomes da tabela para codificar e se houver algum erro no código ou na lógica, o programa irá para de funcionar. Também peguei alguns resultados e comparei com a tabela no próprio Excel e no Data Frame do pandas para verificar se as informações estavam corretas.

NM_BOLSISTA	CPF_BOLSISTA	NM_ENTIDADE_ENSINO	ME_REFERENCIA	AN_REFERENCIA
JULIANA NOVAIS LOPES	***.477.631-**	UNIVERSIDADE DO ESTADO DA BAHIA	1	2013

*Print referente ao terceiro resultado mostrado acima

3. Consultar Média Anual:

Na terceira opção do menu, o programa recebe o ano e verifica se ele está entre o maior e o menor ano contido na tabela. Se for verdade, a função media_ano() é chamada e então localiza todas as linhas em que o ano é igual ao solicitado e calcula a média da coluna com o valor das bolsas através do .mean()

Resultado:

```
Informe o ano que deseja para saber a média das bolsas: 2014
A Média dos valores das bolsas do ano de 2014 é aproximadamente R$ 936.24
```

Testes:

No Excel, selecionei todas as linhas que continham tal ano e então apliquei a função "AVERAGE" que me retornou a média. O Resultado divergiu por muito levemente, porém, acredito que seja pela forma de arredondamento do Excel.

Dessa vez optei por não criar uma função para testar pois sera repetir extamente o que Python/Pandas fazem com o .mean().

4. Rankear as 3 maiores/menores bolsas:

Nesta opção, o programa aplica o comando sort_values() no Data Frame que ordena de acordo com o paramêtro *ascending = True ou False*.

Para pegar somente as 3 maiores ou menores, apliquei o método head(x), sendo x a quantidade de linhas que eu quero que apareça.

Por fim, o programa printa as informações por indexação da tabela.

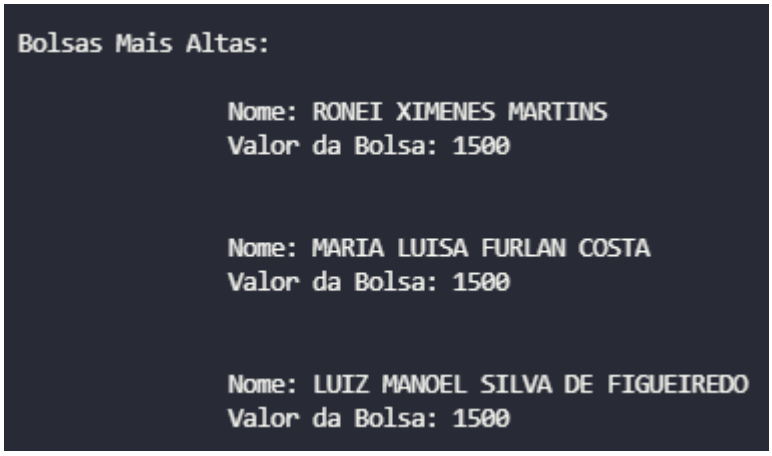
Resultado:

```
Bolsas Mais Baixas:

                Nome: ALEXANDRE RIBEIRO NETO
                Valor da Bolsa: 765

                Nome: JACQUES ALVES DA SILVA
                Valor da Bolsa: 765

                Nome: KELLY PERLIN CASSOL
                Valor da Bolsa: 765
```



Entretanto, notei que existem muito mais pessoas com esses valores de bolsas. Então usei de um comando que conta a quantidade de vezes que cada valor aparece na coluna:

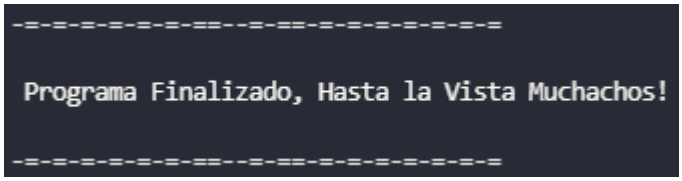
```
765     51247
1300    18549
1100     7501
1400     2048
1500      670
Name: VL_BOLSISTA_PAGAMENTO, dtype: int64
```

Logo, conclui que não há somente três pessoas com as bolsas mais baixas ou mais altas, e sim vários. Então fiquei em dúvida sobre o resultado mostrado pelo programa e optei por printar os três primeiros que o programa encontrasse.

---

5. Finalizar Programa:

Na quinta e última opção, printei uma mensagem e usei de um comando da biblioteca "*os*" para encerrar o programa. Optei por esse comando pois era um dos únicos que não gerava uma exceção para terminar o programa, pois nesse caso, o programa iria cair no except e informar uma mensagem incorreta.



Auto Avaliação:

Com certeza esse programa foi desafiador e cansativo de se executar, porém, no final todo esforço é recompensador. Para começar, gostaria de destacar minha persistência em não desistir e continuar até finalizar sempre procurando a maneira mais eficiente, mas aceitando que talvez o caminho que eu estava seguindo na hora de programar não iria funcionar ou me causaria problemas. Também ressalto que um dos pontos fortes foi a pesquisa, durante todo o processo eu pesquisava comandos novos, jeitos novos de fazer as coisas, sempre olhando as documentações das bibliotecas e do Python para ver se achava algo interessante. Outro ponto forte é que sou muito detalhista, então percebi erros mínimos que não apareciam no print ou atrapalhariam no futuro. Porém, ser detalhista me atrapalhou um pouco, pois perdi muito tempo arrumando coisas bobas que não influenciariam em nada. Tive dificuldade também em seguir meu check-list, que eu fiz justamente para me organizar e não me perder, porém, como estava um pouco ansioso por não saber se iria conseguir terminar, fui fazendo tudo fora de ordem e fiquei perdido, então tive que recomeçar, parar e pensar novamente em algumas em algumas partes.

Obrigado!