dataframe

January 25, 2022

[1]:
```
import pandas as pd
```

[2]:
```
#criando hisóricos de estudantes com seus nomes, cursos e nota
r1 = pd.Series({'Name': 'Alice',
                'Class': 'Física',
                'Score': 85})
r2 = pd.Series({'Name': 'Jack',
                'Class': 'Química',
                'Score': 75})
r3 = pd.Series({'Name': 'Helen',
                'Class': 'Biologia',
                'Score': 90})
```

[5]:
```
#assim como uma série, um objeto de dataframe é index. vamos usar um grupo de␣
 ↪séries,
#onde cada série representa uma linha de dado. iguial uma função de série,␣
 ↪podemos passar nosso item individual
#como um array e podemos também passar nosso valor indexado como segundo␣
 ↪argumento

df = pd.DataFrame([r1, r1, r3], index = ['Escola1', 'Escola2', 'Escola1'])

df.head()
```

[5]:

	Name	Class	Score
Escola1	Alice	Física	85
Escola2	Alice	Física	85
Escola1	Helen	Biologia	90

[8]:
```
#um método alternativo seria usar uma lista de dicionários, onde cada␣
 ↪dicionário representa uma linha de dados

students = [{'Name': 'Alice',
             'Class': 'Física',
             'Score': 85},
            {'Name': 'Jack',
             'Class': 'Química',
             'Score': 75},
            {'Name': 'Helen',
```

```
            'Class': 'Biologia',
            'Score': 90}]
df2 = pd.DataFrame(students, index =['Escola1', 'Escola2', 'Escola1'])
df2.head()
```

[8]:
```
          Name     Class  Score
Escola1  Alice    Física     85
Escola2   Jack   Química     75
Escola1  Helen  Biologia     90
```

[11]:
```
#similar à séries, podemos extrair dados usando os atributos .iloc e .loc
#o dataframe tem duas dimensões, então passando um púnico valor para o .loc,␣
 ↪retornaria
#a série caso houvesse apenas uma linha para retornar

#por exemplo, se quisermos selecionar dados associados com a escola2, fazemos a␣
 ↪query
#com o .loc e um parâmetro
df2.loc['Escola2']
```

[11]:
```
Name        Jack
Class    Química
Score         75
Name: Escola2, dtype: object
```

[14]:
```
#podemos checar o tipo de dados do iloc usando o type
type(df2.loc['Escola2'])
```

[14]:
```
pandas.core.series.Series
```

[15]:
```
#se quisermos listar o nome dos estudantes da escola 1, nós passamos dois␣
 ↪parâmetros para o .loc[]
#um deles sendo o indexador da linha e o outro o nome da coluna

#assim, se estivermos interessados apenas em nomes de alunos da escola 1
df.loc['Escola1', 'Name']
```

[15]:
```
Escola1    Alice
Escola1    Helen
Name: Name, dtype: object
```

[22]:
```
#e o que faríamos para selecionar apenas uma coluna. primeiros temos que␣
 ↪traanspor a matriz,
#isso põe todas as linhas em colunas e todas as colunas em linhas

df2.head()
```

[22]:
```
          Name     Class  Score
Escola1  Alice    Física     85
Escola2   Jack   Química     75
Escola1  Helen  Biologia     90
```

[23]:
```
df2.T
```

[23]:
```
        Escola1  Escola2   Escola1
Name      Alice     Jack     Helen
Class   Física  Química  Biologia
Score       85       75        90
```

[24]:
```
#daí em seguida podemos usar o .loc no transposto para pegar apenas o nome dos␣
 ↪alunos
df.T.loc['Name']
```

[24]:
```
Escola1    Alice
Escola2    Alice
Escola1    Helen
Name: Name, dtype: object
```

[ ]:
```
#como o iloc e loc são usados para seleção de linhas, o pandas reserva um␣
 ↪operador de indexação
#diretamente no DataFrame para seleção de colunas. Num DataFrame do pandas,␣
 ↪colunas sempre tem um nome, então
#essa seleção é sempre baseada em labels e não é tão confusa quanto usar o␣
 ↪colchete em objetos de série
#é parecido com o operador SELECT de projeção de coluna do SQL
```

[25]:
```
df['Name']
```

[25]:
```
Escola1    Alice
Escola2    Alice
Escola1    Helen
Name: Name, dtype: object
```

[26]:
```
#como o resultado de usar o indexador de operação é um dataframe ou série,␣
 ↪podemos encadear operações juntas
#por exemplo, podemos selecionar todas as linhas que se relacionam com escola␣
 ↪usando o operador .loc[]
#e daí projetar o nome da coluna apenas para essas linhas

df.loc['Escola1']['Name']
```

[26]:
```
Escola1    Alice
Escola1    Helen
Name: Name, dtype: object
```

[ ]:
```
#encadear costuma trazer uma cópia do dataframe ao invés de uma visão dele.
#para selecionar dados, isso não é um grande problema, mesmo sendo mais lento␣
 ↪que o necessário.
#mas se estiver atualziando, mudando dados através deste método, isso pode ser␣
 ↪uma distração e uma fonte de erros
```

[32]:
```
#se quisermos selecionar todas as linhas, usamos dois pontos : para indicar um␣
 ↪full slice do começo ao fim
```

```
#isso é como fatiar caracteres numa lista python, daí podemos adicionar o nome␣
 ↪da coluna como um segundo parâmetro na
#forma de sttring. se quisermos incluir multiplas colunas, podemos fazer isso␣
 ↪numa forma de lista e o pandas
#vai trazer de volta somente as colunas que selecionamos

#aqui vai um exemplo onde pedimos todos os nomes e notas de todas as escolas␣
 ↪usando o .loc
df2.loc[:, ['Name', 'Score']]
```

[32]:

	Name	Score
Escola1	Alice	85
Escola2	Jack	75
Escola1	Helen	90

[ ]:
```
#no código acima, os dois pontos pedem que sejam todas as linhas e a lista as␣
 ↪colunas que queremos de volta
```

[36]:
```
#dropping data usando a função drop()
#ele na verdade não muda o dataframe, ele tráz de volta uma cópia do dataframe␣
 ↪com tais colunas dropadas(removidas)
df2.drop('Escola1')
```

[36]:

	Name	Class	Score
Escola2	Jack	Química	75

[37]:
```
df2
```

[37]:

	Name	Class	Score
Escola1	Alice	Física	85
Escola2	Jack	Química	75
Escola1	Helen	Biologia	90

[ ]:
```
#o drop tem dois parâmetos opicionais. o primeiro é chamado inplace, e se␣
 ↪setado para True, o dataframe vai ser atualizado
#ao invés de ser feito uma cópia. o segundo parâmetro são os eixos,que devem␣
 ↪ser dropados.
#como default esse valor é 0, indicado o eixo de linhas, mas podemos trocar␣
 ↪para 1 se desejarmos dropar uma coluna
```

[40]:
```
copy_df = df.copy()
copy_df.drop('Name', inplace = True, axis = 1)
copy_df
```

[40]:

	Class	Score
Escola1	Física	85
Escola2	Física	85
Escola1	Biologia	90

[41]:
```
df
```

[41]:
```
          Name     Class  Score
Escola1  Alice    Física     85
Escola2  Alice    Física     85
Escola1  Helen  Biologia     90
```

[42]:
```
#tem uma segunda forma de dropar uma coluna, e é direto pelo uso do indexador, usando a palavra 'del'
#essa forma de dropar um dado tem efeito imediato no dataframe e não retorna uma view
del copy_df['Class']
copy_df
```

[42]:
```
         Score
Escola1     85
Escola2     85
Escola1     90
```

[44]:
```
#e agora adicionamento uma nova coluna no dataframe.
#podemos adicionar uma coluna como ranking de classe com valor default de None, podemos usar
#o operador de atribuição depois dos colchetes
#isso transmite o valor default para a nova coluna imediatamente

df['ClassRanking'] = None
d
```

[44]:
```
          Name     Class  Score ClassRanking
Escola1  Alice    Física     85         None
Escola2  Alice    Física     85         None
Escola1  Helen  Biologia     90         None
```