TECHMIMO

Autor: Rafael Pereira da Silva

Seguem alguns recados para ajudá-los e para contribuir com o curso:

- Fiquem à vontade para me contatar pelo Linkdin, costumo responder por lá também: https://www.linkedin.com/in/rafael-pereira-da-silva-23890799/ (https://www.linkedin.com/in/rafael-pereira-da-silva-23890799/)
- Fiquem a vontade para compartilharem o certificado do curso no Linkedin. Eu costumo curtir e comentar para dar mais credibilidade
- Vocês podem usar esses notebooks para resolver os exercícios e desafios
- Não se esqueçam de avaliar o curso e dar feedback, eu costumo criar conteúdos baseado nas demandas de vocês
- Se tiverem gostando do curso, recomendem aos amigos, pois isso também ajuda a impulsionar e a crescer a comunidade
- Bons estudos e grande abraços!

Seção 9 - Fundamentos de Pandas

Acesse: https://pandas.pydata.org/docs/user_guide/index.html (https://pandas.pydata.org/docs/user_guide/index.html)

Acesse: https://pandas.pydata.org/docs/user_guide/index.html (https://pandas.pydata.org/docs/user_guide/index.html)

Dados em csv: https://www.kaggle.com/drgilermo/nba-players-stats (https://www.kaggle.com/drgilermo/nba-players-stats)

9.1 - Estrutura de dados no Pandas

Series - *Series* são estruturas de dados 1D e podem ser criadas a partir de dicionários, arrays, entre outros. É muito parecido com os arrays em termos de propriedades e métodos.

- pd.Series(data, index=index)

DataFrame - *DataFrames* são estruturas de dados 2D e podem ser criadas a partir de dicionários, dados estruturados em 2D, arquivos csv, etc. Em resumo, DataFrame é uma **tabela**.

- pd.DataFrame({'key_1': pd.Series(...), 'key_n':pd.Series(...)})
- pd.read_csv(file)

In [57]:

```
# Dados

lista_a = [11,22,33,44,55]
lista_b = ['a','b','c','d','e']

import numpy as np

arr = np.arange(10)**2

dic = {'a':123,'b':456,'c':789}

dic2 = {'coluna1':np.arange(10)*5,'coluna2':np.arange(10)**2,'coluna3':np.ones(10)}
dic2['coluna1']
```

Out[57]:

```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45])
```

In [50]:

```
import pandas as pd


serie_1 = pd.Series(lista_b,index=lista_a)
serie_1[22]
```

Out[50]:

```
'b'
```

In [52]:

```
serie_3 = pd.Series(arr)
serie_3
```

Out[52]:

```
0     0
1     1
2     4
3     9
4    16
5    25
6    36
7    49
8    64
9    81
dtype: int32
```

In [55]:

```
serie_4 = pd.Series(dic)
serie_4['a']
```

Out[55]:

```
123
```

In [61]:

```
df1 = pd.DataFrame(dic2)
type(df1['coluna1'])
```

Out[61]:

```
pandas.core.series.Series
```

In [63]:

```
arr2 = np.arange(15).reshape(5,3)


df3 = pd.DataFrame(arr2)

df3
```

Out[63]:

	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8
3	9	10	11
4	12	13	14

In [64]:

```
file = 'datasets_1358_30676_Players.csv'

NBA_players = pd.read_csv(file)
NBA_players
```

Out[64]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky
...	...	...	...	...	...	...	...	...
3917	3917	Troy Williams	198.0	97.0	South Carolina State University	1969.0	Columbia	South Carolina
3918	3918	Kyle Wiltjer	208.0	108.0	Gonzaga University	1992.0	Portland	Oregon
3919	3919	Stephen Zimmerman	213.0	108.0	University of Nevada, Las Vegas	1996.0	Hendersonville	Tennessee
3920	3920	Paul Zipser	203.0	97.0	NaN	1994.0	Heidelberg	Germany
3921	3921	Ivica Zubac	216.0	120.0	NaN	1997.0	Mostar	Bosnia and Herzegovina

3922 rows × 8 columns

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

9.2 Funcionalidades básicas

Acesse: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)

Algumas propriedades de DataFrames

shape, size, dtypes, columns, index (retorna linhas), values (retorna array)

Alguns métodos para DataFrames

Métodos aplicados ao DataFrame	Descrição
.head(*optativo*)	para ver primeiras linhas
.tail(*optativo*)	para ver últimas linhas
.to_csv(*path/name*)	exporta o DF para um arquivo csv (esse método existe para vários outros formatos)
.min(axis), .max(axis), .cumsum(),mean()	similar as discutidas na seção de Numpy
.value_counts()	Este método faz uma contagem de aparições de determinado valor (aplicar para Series)
.sort_values(by=column)	Ordena valores.

In [68]:

```
import pandas as pd


lista_a = ['a','a','b','c','a','d','c']

lista_b = list(range(7))

dic = {'string':lista_a,'numeros':lista_b}
dic
```

Out[68]:

```
{'string': ['a', 'a', 'b', 'c', 'a', 'd', 'c'],
 'numeros': [0, 1, 2, 3, 4, 5, 6]}
```

In [81]:

```
df = pd.DataFrame(dic)

type(df.values)
```

Out[81]:

```
numpy.ndarray
```

In [88]:

```
import numpy as np

df2 = pd.DataFrame({'col1':np.arange(100)**2,'col2':np.arange(100)*5})
df2.tail()
```

Out[88]:

	col1	col2
95	9025	475
96	9216	480
97	9409	485
98	9604	490
99	9801	495

In [96]:

```
df.cumsum()
```

Out[96]:

	string	numeros
0	a	0
1	aa	1
2	aab	3
3	aabc	6
4	aabca	10
5	aabcad	15
6	aabcadc	21

In [98]:

```
df['string'].value_counts()
```

Out[98]:

```
a    3
c    2
d    1
b    1
Name: string, dtype: int64
```

In [100]:

```
df.sort_values(by='string')
```

Out[100]:

	string	numeros
0	a	0
1	a	1
4	a	4
2	b	2
3	c	3
6	c	6
5	d	5

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

9.3 Tratando dados (Exemplo prático 1)

A partir dos dados de jogadores da NBA (https://www.kaggle.com/drgilermo/nba-players-stats (https://www.kaggle.com/drgilermo/nba-players-stats)), pede-se:

- Qual a média de altura e de peso dos jogadores?
- Crie uma nova coluna com a razão peso/altura.
- Qual a maior altura, o maior peso, a menor altura e menor peso?
- Qual é o estado onde nasce o maior número de jogadores?
- Qual o jogador mais alto?

In [103]:

```
import pandas as pd


file = 'datasets_1358_30676_Players.csv'

df = pd.read_csv(file)

df.head()
```

Out[103]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

In [104]:

```
df.columns
```

Out[104]:

```
Index(['Unnamed: 0', 'Player', 'height', 'weight', 'collage', 'born',
       'birth_city', 'birth_state'],
      dtype='object')
```

In [106]:

```
df['height'].mean()
```

Out[106]:

```
198.70492221372098
```

In [107]:

```
df['weight'].mean()
```

Out[107]:

```
94.78321856669217
```

In [115]:

```
IMC = df['weight']/((df['height']/100)**2)
IMC
```

Out[115]:

```
0       23.765432
1       23.483477
2       23.087868
3       22.907122
4       24.933720
          ...
3917    24.742373
3918    24.963018
3919    23.804801
3920    23.538547
3921    25.720165
Length: 3922, dtype: float64
```

In [116]:

```
df['IMC'] = IMC

df.head()
```

Out[116]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state	razao	
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN	0.427778	2
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana	0.441489	2
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN	0.445596	2
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN	0.448980	2
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky	0.443820	2

In [117]:

```
df.mean()
```

Out[117]:

```
Unnamed: 0    1960.500000
height         198.704922
weight          94.783219
born          1962.379750
razao            0.475704
IMC             23.926953
dtype: float64
```

In [118]:

```
df.min()
```

Out[118]:

```
Unnamed: 0       0.000000
height         160.000000
weight          60.000000
born          1913.000000
razao            0.357143
IMC             16.866251
dtype: float64
```

In [119]:

```
df.max()
```

Out[119]:

```
Unnamed: 0    3921.000000
height         231.000000
weight         163.000000
born          1997.000000
razao            0.721239
IMC             31.913227
dtype: float64
```

In [120]:

```
df['birth_state'].value_counts()
```

Out[120]:

```
California                          344
New York                            290
Illinois                            209
Pennsylvania                        163
Ohio                                137
                                   ...
Saint Vincent and the Grenadines      1
South Africa                          1
Morocco                               1
Cape Verde                            1
New Hampshire                         1
Name: birth_state, Length: 128, dtype: int64
```

In [124]:

```
df.sort_values(by='height').tail(20)
```

Out[124]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state	ra
2547	2547	Zydrunas Ilgauskas	221.0	107.0	NaN	1975.0	Kaunas	Lithuania	0.48
3814	3814	Kristaps Porzingis	221.0	108.0	NaN	1995.0	Liepaja	Latvia	0.48
3381	3381	Hasheem Thabeet	221.0	119.0	University of Connecticut	1987.0	Dar es Salaam	United Republic of Tanzania	0.53
3812	3812	Tibor Pleiss	221.0	116.0	NaN	1989.0	Bergisch Gladbach	Germany	0.52
3822	3822	Walter Tavares	221.0	117.0	NaN	1992.0	Maio	Cape Verde	0.52
1540	1540	Mark Eaton	224.0	124.0	University of California, Los Angeles	1957.0	Westminster	California	0.55
2482	2482	Priest Lauderdale	224.0	147.0	Central State University	1973.0	Chicago	Illinois	0.65
1631	1631	Ralph Sampson*	224.0	103.0	University of Virginia	1960.0	Harrisonburg	Virginia	0.45
1956	1956	Rik Smits	224.0	113.0	Marist College	1966.0	Eindhoven	Netherlands	0.50
3017	3017	Pavel Podkolzin	226.0	117.0	NaN	1985.0	Novosibirsk	Russia	0.51
3018	3018	Peter John	226.0	117.0	NaN	1985.0	NaN	NaN	0.51
2969	2969	Slavko Vranes	226.0	124.0	NaN	1983.0	Belgrade	Serbia	0.54
3686	3686	Sim Bhullar	226.0	163.0	New Mexico State University	1992.0	Ontario	Canada	0.72
1563	1563	Chuck Nevitt	226.0	98.0	North Carolina State University	1959.0	Cortez	Colorado	0.43
2878	2878	Yao Ming*	229.0	140.0	NaN	1980.0	Shanghai	China	0.61
2249	2249	P.J. Brown	229.0	106.0	NaN	1972.0	NaN	NaN	0.46
2248	2248	Shawn Bradley	229.0	106.0	Brigham Young University	1972.0	Landstuhl	Germany	0.46
1711	1711	Manute Bol	231.0	90.0	University of Bridgeport	1962.0	Gogrial	South Sudan	0.38
2297	2297	Gheorghe Muresan	231.0	137.0	NaN	1971.0	Triteni	Romania	0.59
223	223	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [ ]:

```
```

9.4 Fatiando DataFrames com loc e iloc

Acesse: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)

- loc[i,j], onde i e j são os nomes dos índices
- iloc[i,j], onde i e j são os números dos índices

In [161]:

```
import pandas as pd

file = 'datasets_1358_30676_Players.csv'

df = pd.read_csv(file)
df.head()
```

Out[161]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

In [163]:

```
df.columns
```

Out[163]:

```
Index(['Unnamed: 0', 'Player', 'height', 'weight', 'collage', 'born',
       'birth_city', 'birth_state'],
      dtype='object')
```

In [168]:

```
df.loc[1:4,'height':'birth_state']
```

Out[168]:

	height	weight	collage	born	birth_city	birth_state
1	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

In [173]:

```
df.iloc[0:50:3,2:5]
```

Out[173]:

	height	weight	collage
0	180.0	77.0	Indiana University
3	196.0	88.0	North Carolina State University
6	196.0	90.0	University of Kansas
9	196.0	95.0	University of Denver
12	190.0	79.0	Oklahoma State University
15	185.0	81.0	Louisiana State University
18	208.0	106.0	NaN
21	190.0	79.0	Seton Hall University
24	196.0	90.0	East Texas State University
27	190.0	83.0	Louisiana State University
30	188.0	81.0	University of Wisconsin
33	183.0	81.0	Georgetown University
36	185.0	74.0	University of Pennsylvania
39	185.0	79.0	Seton Hall University
42	183.0	79.0	New York University
45	198.0	88.0	College of William & Mary
48	188.0	79.0	Butler University

In [ ]:

```

```

9.5 Função merge e função concat

Acesse: https://pandas.pydata.org/docs/user_guide/merging.html (https://pandas.pydata.org/docs/user_guide/merging.html)

De forma resumida, a função concat nos auxilia em combinar, concatenar os DataFrames. A função merge nos ajuda a fundir os DataFrames, tendo uma funcionalidade parecida com o procv do Excel.

- pandas.concat(objs: Union[Iterable[FrameOrSeries], Mapping[Label, FrameOrSeries]], axis='0', join: str = '"outer"', ignore_index: bool = 'False', keys='None', levels='None', names='None', verify_integrity: bool = 'False', sort: bool = 'False', copy: bool = 'True')

- pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=True, suffixes=('_x', '_y'), copy=True, indicator=False,validate=None)

pd.concat(args)

In [44]:

```
import pandas as pd
import numpy as np

dfa = pd.DataFrame(np.arange(15).reshape(5,3))
dfb = pd.DataFrame((np.arange(9)**2).reshape(3,3))

dfb
```

Out[44]:

	0	1	2
0	0	1	4
1	9	16	25
2	36	49	64

In [46]:

```
pd.concat([dfa,dfb],ignore_index=True)
```

Out[46]:

	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8
3	9	10	11
4	12	13	14
5	0	1	4
6	9	16	25
7	36	49	64

pd.merge(args)

In [51]:

```
df1 = pd.DataFrame({'id':[1,2,3],
                    'cor':['branco','preto','verde']})

df2 = pd.DataFrame({'times':['time1','time2','time3','time4','time5'],
                    'id_cor':[1,1,2,3,2]})

df2
```

Out[51]:

	times	id_cor
0	time1	1
1	time2	1
2	time3	2
3	time4	3
4	time5	2

In [53]:

```
pd.merge(df1,df2,left_on='id',right_on='id_cor')
```

Out[53]:

	id	cor	times	id_cor
0	1	branco	time1	1
1	1	branco	time2	1
2	2	preto	time3	2
3	2	preto	time5	2
4	3	verde	time4	3

9.6 Índices booleanos

Os índices booleanos nos auxiliam a aplicar filtros nos dataframes.

In [56]:

```
import pandas as pd

file = 'datasets_1358_30676_Players.csv'

df = pd.read_csv(file)
df.head(5)
```

Out[56]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

In [57]:

```
df.columns
```

Out[57]:

```
Index(['Unnamed: 0', 'Player', 'height', 'weight', 'collage', 'born',
       'birth_city', 'birth_state'],
      dtype='object')
```

In [58]:

```
df['birth_state'].value_counts()
```

Out[58]:

```
California                    344
New York                      290
Illinois                      209
Pennsylvania                  163
Ohio                          137
                             ...
Estonia                         1
Denmark                         1
United Republic of Tanzania     1
Trinidad and Tobago             1
South Africa                    1
Name: birth_state, Length: 128, dtype: int64
```

In [61]:

```
From_California = df['birth_state'] =='California'

From_California.value_counts()
```

Out[61]:

```
False    3578
True      344
Name: birth_state, dtype: int64
```

In [63]:

```
df[From_California].head(2)
```

Out[63]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
22	22	Bill Calhoun	190.0	81.0	City College of San Francisco	1927.0	San Francisco	California
50	50	Bob Feerick	190.0	86.0	Santa Clara University	1920.0	San Francisco	California

In [65]:

```
after_1970 = df['born']>1970

df[after_1970]
```

Out[65]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
2217	2217	Harold Miner	196.0	95.0	University of Southern California	1971.0	Inglewood	California
2220	2220	Tracy Murray	201.0	102.0	University of California, Los Angeles	1971.0	Los Angeles	California
2222	2222	Shaquille O'Neal*	216.0	147.0	Louisiana State University	1972.0	Newark	New Jersey
2246	2246	Vin Baker	211.0	105.0	University of Hartford	1971.0	Lake Wales	Florida
2248	2248	Shawn Bradley	229.0	106.0	Brigham Young University	1972.0	Landstuhl	Germany
...	...	...	...	...	...	...	...	...
3916	3916	Isaiah Whitehead	193.0	96.0	Seton Hall University	1995.0	Brooklyn	New York
3918	3918	Kyle Wiltjer	208.0	108.0	Gonzaga University	1992.0	Portland	Oregon
3919	3919	Stephen Zimmerman	213.0	108.0	University of Nevada, Las Vegas	1996.0	Hendersonville	Tennessee
3920	3920	Paul Zipser	203.0	97.0	NaN	1994.0	Heidelberg	Germany
3921	3921	Ivica Zubac	216.0	120.0	NaN	1997.0	Mostar	Bosnia and Herzegovina

1487 rows × 8 columns

In [ ]:

```

```

In [ ]:

```

```

9.7 Tabela dinâmica (Tratando carteira de ações)

- pivot_table(values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All', observed=False)

In [73]:

```
import pandas as pd

file = 'Acoes_ficticias.csv'

df = pd.read_csv(file,delimiter=';')
df.head(10)
```

Out[73]:

	Papel	preco_compra	Quantidade
0	PETR4	92.114785	100
1	VVAR3	17.519059	100
2	ITSA4	56.375177	200
3	ITSA4	62.284332	300
4	EMBR3	80.626768	100
5	ITSA4	77.354576	300
6	PETR4	7.330583	100
7	PETR4	37.425152	200
8	EMBR3	62.691443	300
9	VVAR3	58.069377	100

In [72]:

```
df.columns
```

Out[72]:

```
Index(['Papel', 'preco_compra', 'Quantidade'], dtype='object')
```

In [77]:

```
pd.pivot_table(df,index='Papel',values='Quantidade',aggfunc='sum')
```

Out[77]:

	Quantidade
Papel	
EMBR3	900
ITSA4	900
PETR4	1100
VVAR3	800

In [81]:

```
df['valor_total'] = df.preco_compra*df.Quantidade

df.head(2)
```

Out[81]:

	Papel	preco_compra	Quantidade	valor_total
0	PETR4	92.114785	100	9211.478515
1	VVAR3	17.519059	100	1751.905894

In [84]:

```
soma_qtd_valor = pd.pivot_table(df,index='Papel',values=['Quantidade','valor_total'],aggfun

soma_qtd_valor
```

Out[84]:

Papel	Quantidade	valor_total
EMBR3	900	49838.769433
ITSA4	900	61451.615028
PETR4	1100	43671.836644
VVAR3	800	22637.653709

In [85]:

```
soma_qtd_valor['pmedio'] = soma_qtd_valor.valor_total/soma_qtd_valor.Quantidade

soma_qtd_valor
```

Out[85]:

Papel	Quantidade	valor_total	pmedio
EMBR3	900	49838.769433	55.376410
ITSA4	900	61451.615028	68.279572
PETR4	1100	43671.836644	39.701670
VVAR3	800	22637.653709	28.297067

In [86]:

```
pd.pivot_table(df,columns='Papel',values='Quantidade',aggfunc='sum')
```

Out[86]:

Papel	EMBR3	ITSA4	PETR4	VVAR3
Quantidade	900	900	1100	800

In [ ]:

```
```

9.8 Visualização de gráficos com Pandas

Nesta seção veremos:

- df.plot(x='x',y='y')
- df.plot.bar(x='x',y='y')
- df.plot.pie(subplots=True ou y='y')
- df.plot.scatter(x='x',y='y')

Acesse: https://pandas.pydata.org/docs/user_guide/visualization.html#visualization-hist (https://pandas.pydata.org/docs/user_guide/visualization.html#visualization-hist)

Gráfico básico

In [92]:

```
import pandas as pd
import numpy as np

arr = np.arange(10)
arr

serie_1 = pd.Series(arr.cumsum())
serie_1
```

Out[92]:

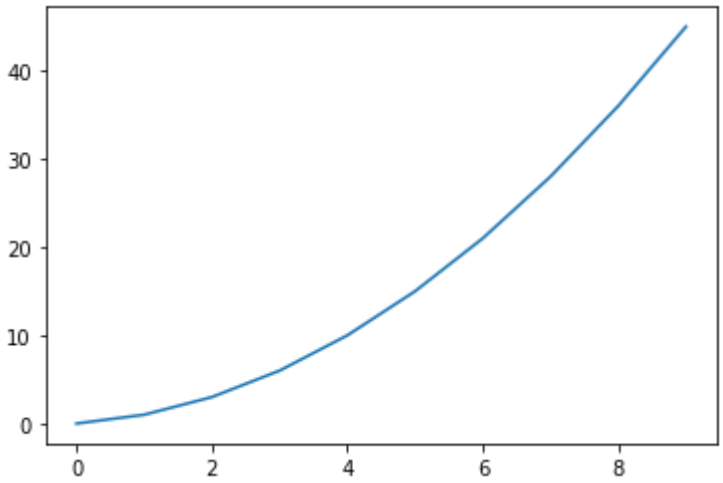
```
0     0
1     1
2     3
3     6
4    10
5    15
6    21
7    28
8    36
9    45
dtype: int32
```

In [93]:

```
serie_1.plot()
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2155c003610>
```



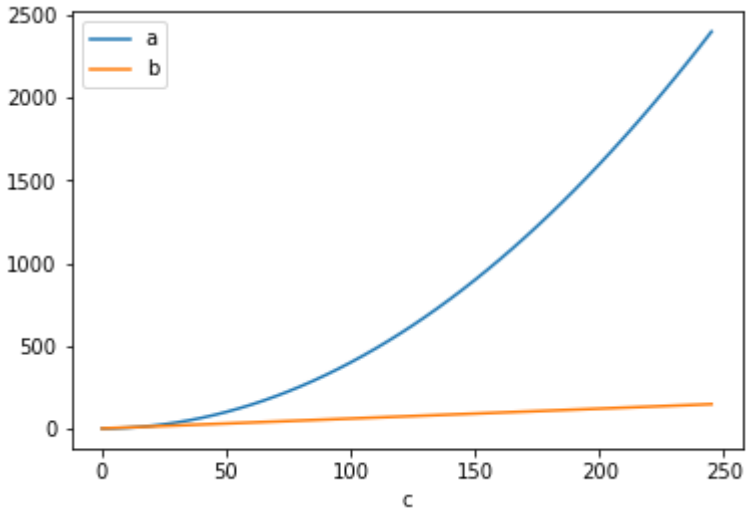
In [97]:

```
df1 = pd.DataFrame({'a':np.arange(50)**2,
                    'b':np.arange(50)*3,
                    'c':np.arange(50)*5})

df1.plot(x='c',y=['a','b'])
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2155b97d190>
```



In [ ]:

```

```

Gráfico de barras

In [98]:

```
soma_qtd_valor
```

Out[98]:

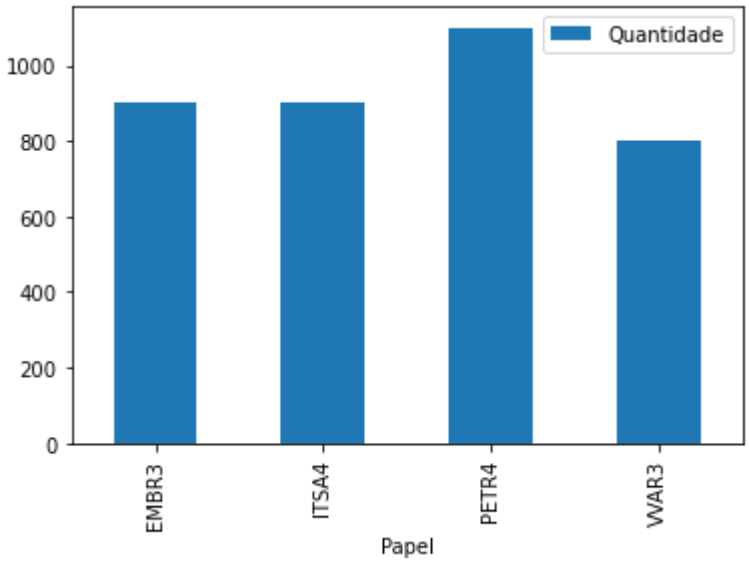
Papel	Quantidade	valor_total	pmedio
EMBR3	900	49838.769433	55.376410
ITSA4	900	61451.615028	68.279572
PETR4	1100	43671.836644	39.701670
VVAR3	800	22637.653709	28.297067

In [102]:

```
soma_qtd_valor.plot.bar(y='Quantidade')
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2155d66ec70>
```



In [ ]:

```

```

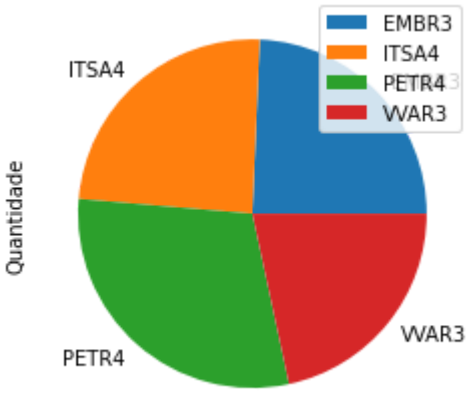
Gráfico de pizza

In [105]:

```
soma_qtd_valor.plot.pie(y='Quantidade')
```

Out[105]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2155d287610>
```



In [ ]:

```

```

Scatter

In [106]:

```
file = 'datasets_1358_30676_Players.csv'

df2 = pd.read_csv(file)

df2.head()
```

Out[106]:

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

In [107]:

```
df2.columns
```

Out[107]:

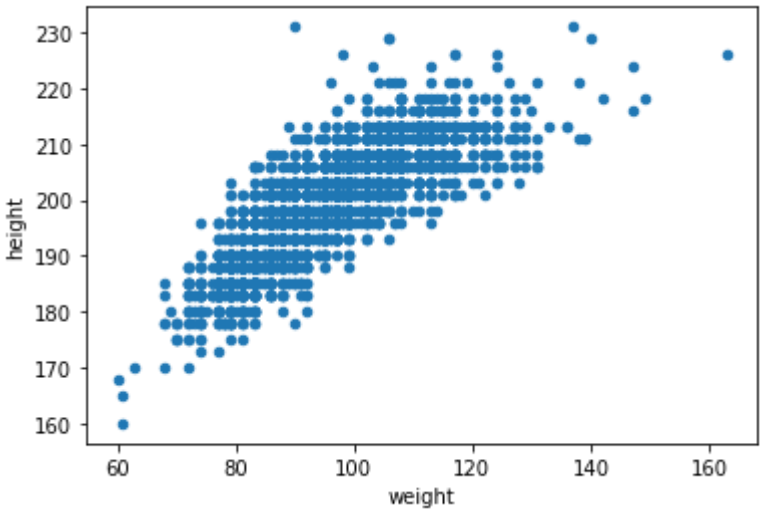
```
Index(['Unnamed: 0', 'Player', 'height', 'weight', 'collage', 'born',
       'birth_city', 'birth_state'],
      dtype='object')
```

In [110]:

```
df2.plot.scatter(x='weight',y='height')
```

Out[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2155d5de610>
```



In [ ]:

```

```

9.9 Tratando dados de intinerários (Exemplo prático 2)

Dados horários de intinerários, pede-se:

- Encontre e filtre as razões sociais que se repetem duas vezes;
- Encontre quais delas possuem intervalos em menos de uma hora.

https://dados.antt.gov.br/dataset/gerenciamento-de-autorizacoes (https://dados.antt.gov.br/dataset/gerenciamento-de-autorizacoes)

Uso do método **.isin** e da função **pd.to_datetime**

In [66]:

```
import pandas as pd

df = pd.read_csv('horarios.csv',sep=';',engine='python')

df.head(2)
```

Out[66]:

	cnpj	razao_social	data_autorizacao	prefixo	descricao_linha	tipo_veiculo	sen
0	04.192.453/0001-18	ALFA LUZ VIACAO TRANSPORTE LTDA	08/07/2016	12-0140-00	BRASILIA(DF) - CALDAS NOVAS(GO)	CONVEN. C/ SANITÁRIO	
1	04.192.453/0001-18	ALFA LUZ VIACAO TRANSPORTE LTDA	08/07/2016	12-0140-00	BRASILIA(DF) - CALDAS NOVAS(GO)	CONVEN. C/ SANITÁRIO	V

2 rows × 27 columns

In [131]:

```
df['time'] = pd.to_datetime(df['horario'])
```

In [132]:

```
df.head(2)
```

Out[132]:

	cnpj	razao_social	data_autorizacao	prefixo	descricao_linha	tipo_veiculo	sen
0	04.192.453/0001-18	ALFA LUZ VIACAO TRANSPORTE LTDA	08/07/2016	12-0140-00	BRASILIA(DF) - CALDAS NOVAS(GO)	CONVEN. C/ SANITÁRIO	
1	04.192.453/0001-18	ALFA LUZ VIACAO TRANSPORTE LTDA	08/07/2016	12-0140-00	BRASILIA(DF) - CALDAS NOVAS(GO)	CONVEN. C/ SANITÁRIO	V

2 rows × 28 columns

In [121]:

```
df.head(2)
```

Out[121]:

	cnpj	razao_social	data_autorizacao	prefixo	descricao_linha	tipo_veiculo	sen
0	04.192.453/0001-18	ALFA LUZ VIACAO TRANSPORTE LTDA	08/07/2016	12-0140-00	BRASILIA(DF) - CALDAS NOVAS(GO)	CONVEN. C/ SANITÁRIO	
1	04.192.453/0001-18	ALFA LUZ VIACAO TRANSPORTE LTDA	08/07/2016	12-0140-00	BRASILIA(DF) - CALDAS NOVAS(GO)	CONVEN. C/ SANITÁRIO	V

2 rows × 28 columns

In [122]:

```
filtro_2 = df['razao_social'].value_counts() == 2
filtro_2
```

Out[122]:

```
OSVALDO MENDES & CIA LTDA. - EMPRESA DOIS IRMÃOS                False
EMPRESA GONTIJO DE TRANSPORTES LTDA.                            False
VIAÇÃO CAIÇARA LTDA.                                            False
CONSÓRCIO GUANABARA DE TRANSPORTES                              False
AUTO VIAÇÃO CATARINENSE LTDA.                                   False
                                                                ...
IRMÃOS NASCIMENTO TURISMO LTDA.                                  True
IVAIR CAETANO DO NASCIMENTO ? ME                                 True
EXPRESSO BRASILEIRO TRANSPORTE RODOVIARIO E TURISMO LTDA - EPP   True
VIAÇÃO MINEIROS TRANSPORTE E TURISMO LTDA                        True
VIAÇÃO PARANAIBA LTDA.                                          False
Name: razao_social, Length: 208, dtype: bool
```

In [123]:

```
lista_filtrada = list(filtro_2[filtro_2].index)
len(lista_filtrada)
```

Out[123]:

```
16
```

In [124]:

```
tabela_filtrada = df[df['razao_social'].isin(lista_filtrada)]
tabela_filtrada
```

Out[124]:

	cnpj	razao_social	data_autorizacao	prefixo	descricao_linha	tipo_veiculo	sentido
1792	04.801.028/0001-89	DANISTUR TRANSPORTE RODOVIARIO LTDA	22/04/2020	12-0489-60	GOIANIA(GO) - XINGUARA(PA)	EXECUTIVO	ida
1793	04.801.028/0001-89	DANISTUR TRANSPORTE RODOVIARIO LTDA	22/04/2020	12-0489-60	GOIANIA(GO) - XINGUARA(PA)	EXECUTIVO	volta
2934	76.423.366/0001-35	EMPRESA TESTE SUPAS	22/09/2020	06-1863-00	PIAU(MG) - AGUA COMPRIDA(MG)	CONVEN. C/ SANITÁRIO	ida
2935	76.423.366/0001-35	EMPRESA TESTE SUPAS	22/09/2020	06-1863-00	PIAU(MG) - AGUA COMPRIDA(MG)	CONVEN. C/ SANITÁRIO	volta
		EXPRESSO					

In [142]:

```
Serie_maxB = pd.pivot_table(tabela_filtrada,index='razao_social',values = ['time'],aggfunc=
Serie_maxB
```

Out[142]:

razao_social	time
DANISTUR TRANSPORTE RODOVIARIO LTDA	2020-10-14 14:00:00
EMPRESA TESTE SUPAS	2020-10-14 14:00:00
EXPRESSO BRASILEIRO TRANSPORTE RODOVIARIO E TURISMO LTDA - EPP	2020-10-14 17:30:00
EXPRESSO METRÓPOLIS TRANSPORTES E VIAGENS LTDA.	2020-10-14 17:15:00
EXPRESSO SANTA MARTA LTDA.	2020-10-14 15:30:00
IRMÃOS NASCIMENTO TURISMO LTDA.	2020-10-14 07:00:00
IVAIR CAETANO DO NASCIMENTO ? ME	2020-10-14 09:00:00
JS SERVIÇOS LOGISTICOS LTDA	2020-10-14 06:00:00
KAWAGUCHI EVENTOS TRANSPORTE E TURISMO LTDA. (CATEDRAL TURISMO)	2020-10-14 12:00:00
ROTA DO SOL TRANSPORTE E TURISMO LTDA	2020-10-14 00:00:00
TPC TRANSPORTES LTDA - ME	2020-10-14 05:00:00
TRANSPEN TRANSPORTE COLETIVO E ENCOMENDAS LTDA.	2020-10-14 08:30:00
VIACAO JLS LTDA	2020-10-14 12:00:00
VIAÇÃO LUXOR LTDA - LLC TRANSPORTES	2020-10-14 22:30:00
VIAÇÃO MINEIROS TRANSPORTE E TURISMO LTDA	2020-10-14 10:00:00
VIAÇÃO TERESÓPOLIS E TURISMO LTDA.	2020-10-14 17:00:00

In [145]:

```
Serie_minB = pd.pivot_table(tabela_filtrada,index='razao_social',values=['time'],aggfunc='m
Serie_minB
```

Out[145]:

razao_social	time
DANISTUR TRANSPORTE RODOVIARIO LTDA	2020-10-14 09:30:00
EMPRESA TESTE SUPAS	2020-10-14 13:00:00
EXPRESSO BRASILEIRO TRANSPORTE RODOVIARIO E TURISMO LTDA - EPP	2020-10-14 10:30:00
EXPRESSO METRÓPOLIS TRANSPORTES E VIAGENS LTDA.	2020-10-14 07:00:00
EXPRESSO SANTA MARTA LTDA.	2020-10-14 10:30:00
IRMÃOS NASCIMENTO TURISMO LTDA.	2020-10-14 07:00:00
IVAIR CAETANO DO NASCIMENTO ? ME	2020-10-14 08:00:00
JS SERVIÇOS LOGISTICOS LTDA	2020-10-14 06:00:00
KAWAGUCHI EVENTOS TRANSPORTE E TURISMO LTDA. (CATEDRAL TURISMO)	2020-10-14 06:00:00
ROTA DO SOL TRANSPORTE E TURISMO LTDA	2020-10-14 00:00:00
TPC TRANSPORTES LTDA - ME	2020-10-14 04:30:00
TRANSPEN TRANSPORTE COLETIVO E ENCOMENDAS LTDA.	2020-10-14 08:15:00
VIACAO JLS LTDA	2020-10-14 12:00:00
VIAÇÃO LUXOR LTDA - LLC TRANSPORTES	2020-10-14 22:30:00
VIAÇÃO MINEIROS TRANSPORTE E TURISMO LTDA	2020-10-14 04:50:00
VIAÇÃO TERESÓPOLIS E TURISMO LTDA.	2020-10-14 07:00:00

In [144]:

```
Serie_maxB - Serie_minB
```

Out[144]:

razao_social	time
DANISTUR TRANSPORTE RODOVIARIO LTDA	04:30:00
EMPRESA TESTE SUPAS	01:00:00
EXPRESSO BRASILEIRO TRANSPORTE RODOVIARIO E TURISMO LTDA - EPP	07:00:00
EXPRESSO METRÓPOLIS TRANSPORTES E VIAGENS LTDA.	10:15:00
EXPRESSO SANTA MARTA LTDA.	05:00:00
IRMÃOS NASCIMENTO TURISMO LTDA.	00:00:00
IVAIR CAETANO DO NASCIMENTO ? ME	01:00:00
JS SERVIÇOS LOGISTICOS LTDA	00:00:00
KAWAGUCHI EVENTOS TRANSPORTE E TURISMO LTDA. (CATEDRAL TURISMO)	06:00:00
ROTA DO SOL TRANSPORTE E TURISMO LTDA	00:00:00
TPC TRANSPORTES LTDA - ME	00:30:00
TRANSPEN TRANSPORTE COLETIVO E ENCOMENDAS LTDA.	00:15:00
VIACAO JLS LTDA	00:00:00
VIAÇÃO LUXOR LTDA - LLC TRANSPORTES	00:00:00
VIAÇÃO MINEIROS TRANSPORTE E TURISMO LTDA	05:10:00
VIAÇÃO TERESÓPOLIS E TURISMO LTDA.	10:00:00

9.10 - Preparando os dados

Alguns métodos para DataFrames

Métodos aplicados ao DataFrame	Descrição
.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')	Remove uma série de dados especificada
.isnull(obj)/.notnull(obj)	cria uma série de booleanos
.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)	Deleta linha (axis=0) ou coluna (axis=1) com célula(s) nula(s)
.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)	Substitui o valor nulo por um valor determinado
.duplicated(subset=None, keep='first')	retorna um booleano com valores duplicados
.drop_duplicates(subset=None, keep='first', inplace=False, ignore_index=False)	deleta linhas com valores duplicados. Pode selecionar uma determinada coluna usando subset

Atenção, esses métodos não modificam o DataFrame

In [42]:

```
## Drop
```

In [49]:

```
import numpy as np
import pandas as pd

a_arr = np.arange(20).reshape(4,5)
a_arr
```

Out[49]:

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [50]:

```
df_columns = ['A','B','C','D','E']
a_df = pd.DataFrame(a_arr,columns=df_columns)
a_df.drop(index=1,columns='E')
```

Out[50]:

	A	B	C	D
0	0	1	2	3
2	10	11	12	13
3	15	16	17	18

In [61]:

```
## isnull notnull
```

In [52]:

```
a_arr = np.vstack([a_arr,np.array([np.nan,np.nan,np.nan,np.nan,np.nan])])
a_df = pd.DataFrame(a_arr)
a_df
```

Out[52]:

	0	1	2	3	4
0	0.0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0
2	10.0	11.0	12.0	13.0	14.0
3	15.0	16.0	17.0	18.0	19.0
4	NaN	NaN	NaN	NaN	NaN

In [60]:

```
a_df[a_df[0].notnull()]
```

Out[60]:

	0	1	2	3	4
0	0.0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0
2	10.0	11.0	12.0	13.0	14.0
3	15.0	16.0	17.0	18.0	19.0

In [62]:

```
## .dropna() .fillna()
```

In [105]:

```
b = np.arange(20,dtype=float).reshape(4,5)
b[0,0] = np.nan

b_df = pd.DataFrame(b)
b_df
```

Out[105]:

	0	1	2	3	4
0	NaN	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0
2	10.0	11.0	12.0	13.0	14.0
3	15.0	16.0	17.0	18.0	19.0

In [106]:

```
b_df.dropna(axis=1,how='any')
```

Out[106]:

	1	2	3	4
0	1.0	2.0	3.0	4.0
1	6.0	7.0	8.0	9.0
2	11.0	12.0	13.0	14.0
3	16.0	17.0	18.0	19.0

In [107]:

```
b_df.fillna(9999)
```

Out[107]:

	0	1	2	3	4
0	9999.0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0	9.0
2	10.0	11.0	12.0	13.0	14.0
3	15.0	16.0	17.0	18.0	19.0

In [108]:

```
## duplicated e drop_duplicates()
```

In [116]:

```
b_df[0][0] = 5.
b_df[0].duplicated()
```

Out[116]:

```
0    False
1     True
2    False
3    False
Name: 0, dtype: bool
```

9.11 Multi índices (ex. com pyNastran)

Objetos multi índices:

MultiIndex.from_tuples(tuples, sortorder=None, names=None)

MultiIndex.from_arrays(arrays, sortorder=None, names=

MultiIndex.from_frame(df, sortorder=None, names=None)

In [2]:

```
import numpy as np

a_arr = np.arange(10).reshape(5,2)
a_arr
```

Out[2]:

```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

In [14]:

```
a_index = pd.MultiIndex.from_tuples([('A',1),('A',2),('B',1),('B',2),('B',3)],names=['first
type(a_index)
```

Out[14]:

```
pandas.core.indexes.multi.MultiIndex
```

In [15]:

```
import pandas as pd

a_df = pd.DataFrame(a_arr,index=a_index)
a_df
```

Out[15]:

			0	1
first	**second**			
A		**1**	0	1
		2	2	3
B		**1**	4	5
		2	6	7
		3	8	9

In [33]:

```
a_df[1]['A']
```

Out[33]:

```
second
1    1
2    3
Name: 1, dtype: int32
```

In [ ]:

```

```

In [119]:

```
## OP2
```

In [120]:

```
from pyNastran.op2.op2 import OP2
op2 = OP2()
```

In [121]:

```
path = 'D:/DESKTOP-2020/PROJETOS_PESSOAIS/Python/1-Python para Engenheiros e Cientistas/Aul
op2.read_op2(path)
```

DEBUG: op2.py:542 combine=True

DEBUG: op2.py:543 -------- reading op2 with read_mode=1 (array sizing) --------

INFO: op2_scalar.py:1556 op2_filename = 'D:/DESKTOP-2020/PROJETOS_PESSOAIS/Python/1-Python para Engenheiros e Cientistas/Aulas editadas/Secao-9/model-001.op2'

DEBUG: op2_reader.py:270 mode = 'optistruct'

DEBUG: op2_scalar.py:1735 table_name=b'OUGV1'

DEBUG: op2_scalar.py:1735 table_name=b'OES1X'

DEBUG: op2.py:562 -------- reading op2 with read_mode=2 (array filling) --------

DEBUG: op2_reader.py:270 mode = 'optistruct'

DEBUG: op2_scalar.py:1735 table_name=b'OUGV1'

DEBUG: op2_scalar.py:1735 table_name=b'OES1X'

DEBUG: op2.py:859 combine_results

DEBUG: op2.py:575 finished reading op2

In [123]:

```
print(op2.get_op2_stats())
```

```
displacements[1]
  isubcase = 1
  type=RealDisplacementArray nnodes=858, table_name=OUGV1
  data: [t1, t2, t3, r1, r2, r3] shape=[1, 858, 6] dtype=float32
  node_gridtype.shape = (858, 2)
  sort1
  lsdvmns = [1]

ctria3_stress[1]
  type=RealPlateStressArray nelements=4 nnodes_per_element=1 nlayers=2 ntota
l=8
  data: [1, ntotal, 8] where 8=[fiber_distance, oxx, oyy, txy, angle, omax,
omin, von_mises]
  element_node.shape = (8, 2)
  data.shape=(1, 8, 8)
  element type: CTRIA3
  s_code: 1
  sort1
  lsdvmns = [1]

cquad4_stress[1]
  type=RealPlateStressArray nelements=784 nnodes_per_element=1 nlayers=2 nto
tal=1568
  data: [1, ntotal, 8] where 8=[fiber_distance, oxx, oyy, txy, angle, omax,
omin, von_mises]
  element_node.shape = (1568, 2)
  data.shape=(1, 1568, 8)
  element type: CQUAD4
  s_code: 1
  sort1
  lsdvmns = [1]
```

In [124]:

```
op2.cquad4_stress[1].data_frame
```

Out[124]:

			index	fiber_distance	oxx	oyy	txy	angle
ElementID	**NodeID**	**Location**						
1	**CEN**	**Top**	0	-0.5	0.034607	1.319078	0.194984	81.555817
		Bottom	1	0.5	0.034607	1.319078	0.194984	81.555817
3	**CEN**	**Top**	2	-0.5	2.077029	0.235260	-0.032872	-1.022184
		Bottom	3	0.5	2.077029	0.235260	-0.032872	-1.022184
4	**CEN**	**Top**	4	-0.5	-0.169760	1.211231	0.141299	84.217476
...	**...**	**...**	...	...	...	...	...	...
786	**CEN**	**Bottom**	1563	0.5	-0.000088	1.000032	-0.000643	-89.963158
787	**CEN**	**Top**	1564	-0.5	0.001810	0.999950	-0.000374	-89.978500
		Bottom	1565	0.5	0.001810	0.999950	-0.000374	-89.978500
788	**CEN**	**Top**	1566	-0.5	-0.000035	0.999970	-0.000262	-89.984970
		Bottom	1567	0.5	-0.000035	0.999970	-0.000262	-89.984970

1568 rows × 9 columns

In [125]:

```
op2.cquad4_stress[1].data_frame.index
```

Out[125]:

```
MultiIndex([(  1, 'CEN',    'Top'),
            (  1, 'CEN', 'Bottom'),
            (  3, 'CEN',    'Top'),
            (  3, 'CEN', 'Bottom'),
            (  4, 'CEN',    'Top'),
            (  4, 'CEN', 'Bottom'),
            (  5, 'CEN',    'Top'),
            (  5, 'CEN', 'Bottom'),
            (  6, 'CEN',    'Top'),
            (  6, 'CEN', 'Bottom'),
            ...
            (784, 'CEN',    'Top'),
            (784, 'CEN', 'Bottom'),
            (785, 'CEN',    'Top'),
            (785, 'CEN', 'Bottom'),
            (786, 'CEN',    'Top'),
            (786, 'CEN', 'Bottom'),
            (787, 'CEN',    'Top'),
            (787, 'CEN', 'Bottom'),
            (788, 'CEN',    'Top'),
            (788, 'CEN', 'Bottom')],
           names=['ElementID', 'NodeID', 'Location'], length=1568)
```

In [ ]:

```
```

In [ ]:

```
```