DataFrame

DataFrames do Pandas

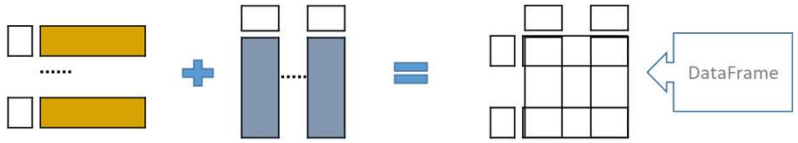
DataFrames: estrutura bidimensional indexada que armazena valores de qualquer tipo.

Índice e Coluna Padrões

	0	1	2	4	4
0	216	Huguinho	9.6	3.8	1.0
1	233	Lalá	6.2	6.9	9.2
2	234	Lelé	6.5	2.7	3.0
3	235	Lili	1.3	4.6	6.5
4	230	Luisinho	9.8	3.7	9.3
5	215	Zezinho	6.2	1.3	8.5

Índice e Coluna Dado

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

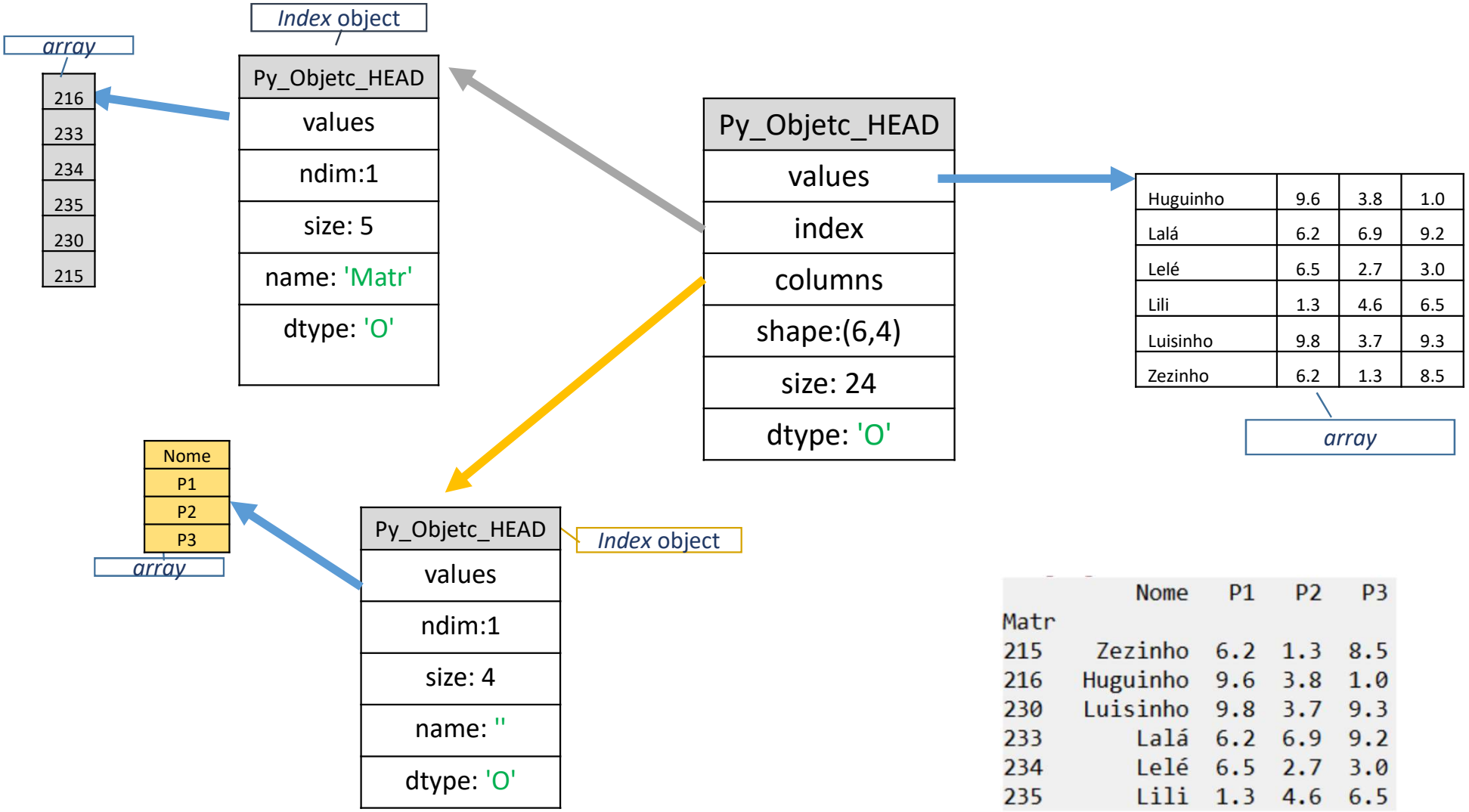


Estrutura tabular com linhas e colunas, similar a uma planilha, composta por:

- **valores**: *array* bidimensional (estruturado ou homogêneo), dicionário (que pode conter df, arrays, constants ou objetos do tipo lista) ou DataFrame
- **índices**: uma sequência de números ou rótulos (*labels)* quaisquer que identificam as linhas
- **colunas**: uma sequência de números ou rótulos (*labels*) quaisquer que identificam as colunas

✓ *Os índices e as colunas não precisam ser exclusivos. Por padrão, variam de 0 a itens -1*

Esquema simplificado do objeto DataFrame



	Nome	P1	P2	P3
Matr				
215	Zezinho	6.2	1.3	8.5
216	Huguinho	9.6	3.8	1.0
230	Luisinho	9.8	3.7	9.3
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

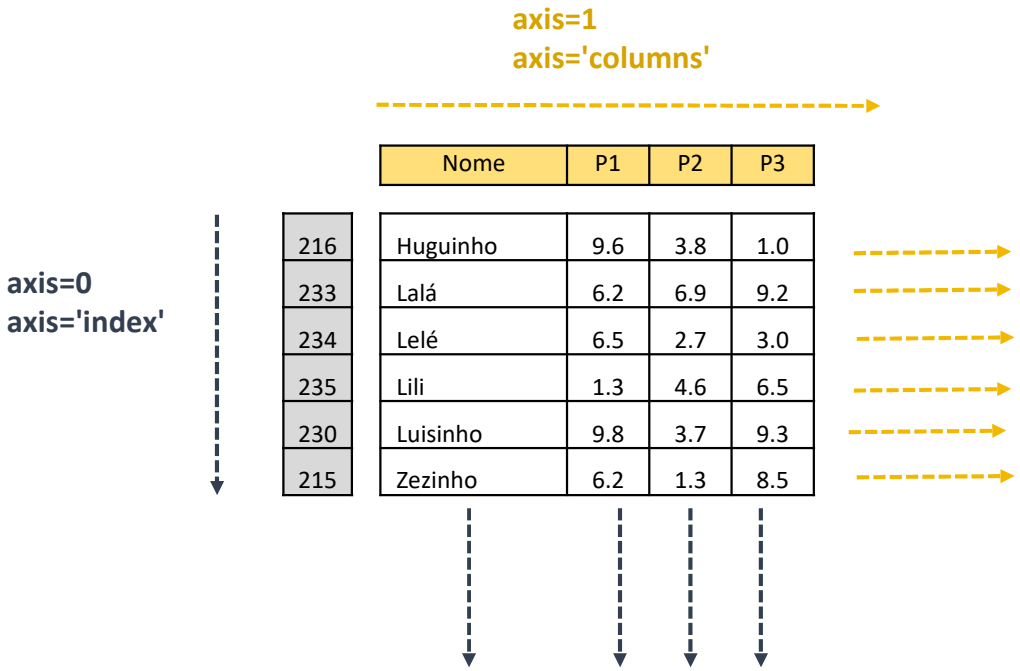
Atributos e Exibição

Valores	`df.values`
Formato	`df.shape`
qt de valores	`df.size`
Transposta	`df.T`
Labels dos Índices	`df.index`
Nome do obj index	`df.index.name`
Labels das colunas	`df.columns`
Nome do obj columns	`df.columns.name`
Inf. da estrutura	`df.info()`
Primeiros Elementos	`df.head(n)  default, n=5`
Últimos Elementos	`df.tail(n) default, n=5`
Resumos Estatísticos das colunas numéricas	`df.describe()`

DataFrames do Pandas

Estrutura tabular com linhas e colunas, similar a uma planilha, composta por:

- ✓ *"grupo de Series que compartilham um índice (nome das colunas)" ou*
- ✓ *"um dicionário de Series"*



Alterando nome de um eixo (lista de colunas/linhas)

Sintaxe: `df.rename_axis(mapper, axis=0, inplace=False)`

Altera o nome do *index* ou *columns*. Retorna um *DataFrame* com o nome da lista de colunas/linhas alterado

mapper - valor que será o nome do eixo (escalar, list-like)

axis = *n/nome* – eixo a renomear. Padrão: 0 (altera o nome do *index*). Pode ser o nome do eixo

inplace = False/True. Padrão: False. Com inplace=True, realiza a operação no DataFrame, não cria uma cópia.

df:

Nome	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

```
df.rename_axis("NOME",inplace=True)
# equivalente a:
#    dfNtd.rename_axis("NOME",axis="index",inplace=True)
#    dfNtd.index.name='NOME'
```

NOME	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

```
df.rename_axis("AL",axis=1,inplace=True)
# equivalente a:
#    dfNtd.rename_axis("AL",axis="columns",inplace=True)
#    dfNtd.columns.name='AL'
```

AL	Matr	P1	P2	P3
NOME				
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

Construindo DataFrames

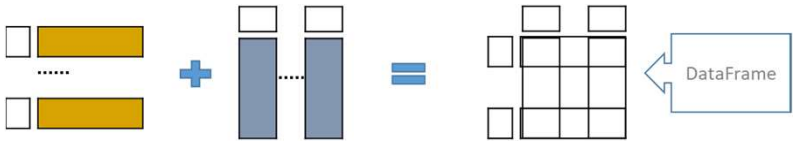
```
pd.DataFrame(...)
```

DataFrames do Pandas

DataFrames: estrutura bidimensional indexada que armazena valores de qualquer tipo.

	0	1	2	4	4
0	216	Huguinho	9.6	3.8	1.0
1	233	Lalá	6.2	6.9	9.2
2	234	Lelé	6.5	2.7	3.0
3	235	Lili	1.3	4.6	6.5
4	230	Luisinho	9.8	3.7	9.3
5	215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5



Estrutura tabular com linhas e colunas, similar a uma planilha, composta por:

- **valores**: *array* bidimensional (estruturado ou homogêneo), dicionário (que pode conter Series, arrays, constants ou objetos do tipo lista) ou DataFrame
- **índices**: uma sequência de números ou rótulos (*labels)* quaisquer que identificam as linhas
- **colunas**: uma sequência de números ou rótulos (*labels*) quaisquer que identificam as colunas

✓ *Os índices e as colunas não precisam ser exclusivos. Por padrão, variam de 0 a itens -1*

Construindo um DataFrame

sem especificação dos índices

Sintaxe:

```
pd.DataFrame(valores)
```

a partir de uma lista de listas

```
>>>lNtA = [
   ['Lalá',133,6.2,6.9,9.2],
   ['Lelé',131,6.5,2.7]]

>>>dfNtA= pd.DataFrame(lNtA)
>>>dfNtA
```

	0	1	2	3	4
0	Lalá	133	6.2	6.9	9.2
1	Lelé	131	6.5	2.7	NaN

a partir do dicionário de listas

```
>>>dNtB= {
   'Lalá':[133,6.2,6.9,9.2],
   'Lelé':[131,6.5,2.7,None]}

>>>dfNtB= pd.DataFrame(dNtB)
>>>dfNtB
```

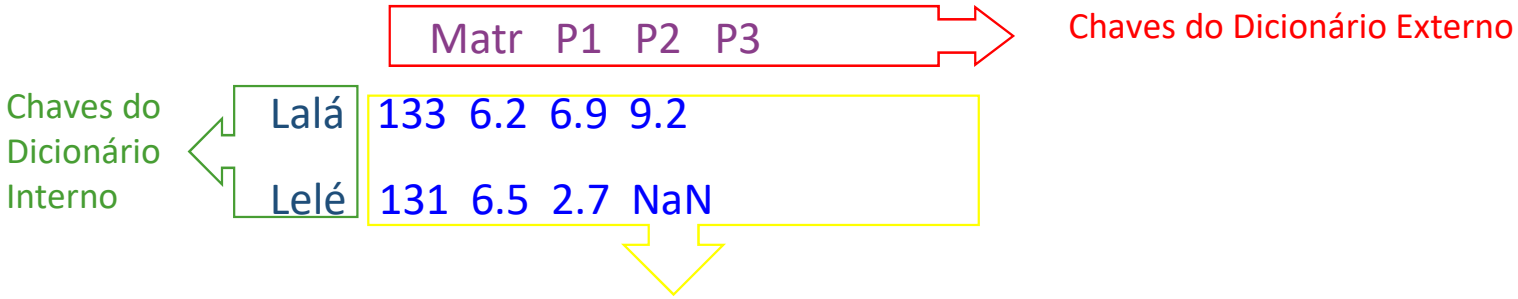
	Lalá	Lelé
0	133.0	131.0
1	6.2	6.5
2	6.9	2.7
3	9.2	NaN

Construindo um DataFrame

a partir de dicionário de dicionários$_{1/2}$

```
# a partir de dicionário de dicionários
dNtC= {'Matr':{'Lalá': 133, 'Lelé': 131},
       'P1': {'Lalá': 6.2, 'Lelé': 6.5},
       'P2': {'Lelé': 2.7, 'Lalá': 6.9},
       'P3': {'Lalá': 9.2}          }

>>>dfNtC= pd.DataFrame(dNtC)
>>>dfNtC          #casamento pelas chaves
```



Construindo um DataFrame

a partir de dicionário de dicionários$_{1/2}$

PASSO A PASSO – Casamento pelas chaves

```
# a partir de dicionário de dicionários
dNtC={'Matr': {'Lalá':133, 'Lelé': 131},
      'P1':   {'Lalá':6.2, 'Lelé': 6.5},
      'P2':   {'Lelé':2.7, 'Lalá': 6.9},
      'P3':   {'Lalá':9.2}              }

>>>dfNtC= pd.DataFrame(dNtC)

>>>dfNtC    #casamento pelas chaves
        Matr   P1   P2   P3
Lalá     133  6.2  6.9  9.2
Lelé     131  6.5  2.7  NaN
```

Construindo um DataFrame

a partir de dicionário de dicionários 2/2

```
# a partir de dicionário de dicionários
dNtD= {  'Lalá':{'Matr': 133,'P1': 6.2,'P2': 6.9,'P3': 9.2 },
         'Lelé':{'Matr': 131,'P1': 6.5,'P2': 2.7        }
        }

>>>dfNtD= pd.DataFrame(dNtD)

>>>dfNtD          #casamento pela chave
```

	Lalá	Lelé
Matr	133.0	131.0
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	NaN

Construindo um DataFrame

a partir de dicionário de Series 1/2

```
# a partir de dicionário de Series, index indefinido
sNomes=pd.Series(['Lalá','Lelé'])
sMatrs=pd.Series([133,131])
sP1s=pd.Series([6.2,6.5])
sP2s=pd.Series([6.9,2.7])
sP3s=pd.Series([9.2])

dNtSs={'Nome':sNomes,
       'Matr':sMatrs,
       'P1':sP1s,
       'P2':sP2s,
       'P3':sP3s}
```

Nome :	Matr :	P1 :	P2 :	P3 :
0 Lalá	0 133	0 6.2	0 6.9	0 9.2
1 Lelé	1 131	1 6.5	1 2.7	

```
dfNtSs= pd.DataFrame(dNtSs)

>>>dfNtSs  #casamento pelo index
```

	Matr	Nome	P1	P2	P3
0	133	Lalá	6.2	6.9	9.2
1	131	Lelé	6.5	2.7	NaN

Construindo um DataFrame

a partir de dicionário de Series 2/2

```
# a partir de dicionário de Series, index definido
I=['Lalá','Lelé]
sMatri=pd.Series([133,131],
                 index=I)
sP1i=pd.Series([6.2,6.5],index=I)
sP2i=pd.Series([6.9,2.7],index=I)
sP3i=pd.Series([9.2,3.0],index=I)

dNtSi={'Matr':sMatri,
       'P1':sP1i,
       'P2':sP2i,
       'P3':sP3i}
```

Matr :	P1 :	P2 :	P3 :
Lalá 133	Lalá 6.2	Lalá 6.9	Lalá 9.2
Lelé 131	Lelé 6.5	Lelé 2.7	Lelé: 3.0

```
dfNtSi= pd.DataFrame(dNtSi)

>>>dfNtSi  #casamento pelo index
      Matr   P1   P2   P3
Lalá   133  6.2  6.9  9.2
Lelé   131  6.5  2.7  3.0
```

Construindo um DataFrame

com especificação de índices e/ou colunas

Sintaxe:

```
pd.DataFrame(valores, index=array1d, columns=array1d)
```

```
# a partir de uma lista
lNtA=[['Lalá',133,6.2,6.9,9.2], ['Lelé',131,6.5,2.7,3.0]]
ind = ['Lalá','Lelé']
cols = ['Nome','Matr','P1','P2','P3']

>>>dfNtAic= pd.DataFrame( lNtA , index=ind, columns=cols)
>>> dfNtAic
      Nome   Matr   P1   P2   P3
Lalá  Lalá    133  6.2  6.9  9.2
Lelé  Lelé    131  6.5  2.7  3.0

>>>dfNtAc= pd.DataFrame( lNtA ,columns=cols)
>>> dfNtAc
   Nome  Matr   P1   P2   P3
0  Lalá   133  6.2  6.9  9.2
1  Lelé   131  6.5  2.7  3.0
```

Construindo um DataFrame

com 1ª planilha de um arquivo Excel

Sintaxe:

```
pd.read_excel(caminho, index_col= n, header= n)
```

caminho - localização do arquivo: composto pelo caminho (absoluto/relativo) e nome

index_col = *n* – O número da coluna do arquivo a ser usada como *labels* das linhas (índice). O padrão é None (o arquivo não possui tal coluna)

header = n - O número da linha para os *labels* das colunas (padrão é 0) ou None quando não há tal linha

decimal = ',' - quando o separador de casas decimais é a vírgula, Padrão: '.'

```
dfNtA=pd.read_excel("als.xlsx")
print(dfNtA.head(2))
```

```
      Nome  Matr   P1   P2   P3
0     Lalá   133  6.2  6.9  9.2
1     Lelé   131  6.5  2.7  3.0
```

als.xlsx

	A	B	C	D	E
1	Nome	Matr	P1	P2	P3
2	Lalá	133	6,2	6,9	9,2
3	Lelé	131	6,5	2,7	3,0

Arquivo na mesma pasta do arq .py

Construindo um DataFrame

com 1ª planilha de um arquivo Excel

```
dfNt2=pd.read_excel("als.xlsx",index_col=0)
print(dfNt2.head(2))
```

Nome	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

als.xlsx

	A	B	C	D	E
1	Nome	Matr	P1	P2	P3
2	Lalá	133	6,2	6,9	9,2
3	Lelé	131	6,5	2,7	3,0

```
dfNt3=pd.read_excel("als.xlsx",index_col=1')
print(dfNt3.head(2))
```

Matr	Nome	P1	P2	P3
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0

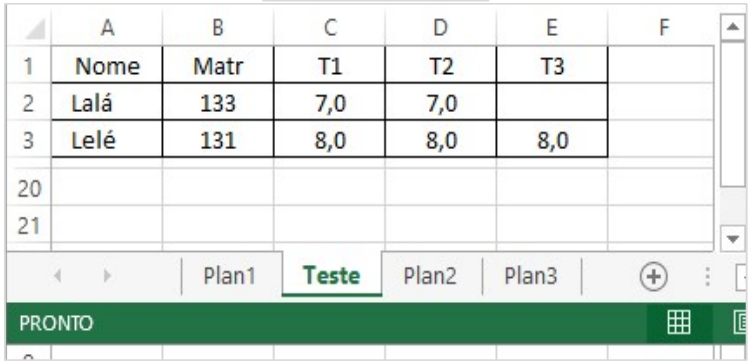
Construindo um DataFrame

com planilha *nomeada* de um arquivo Excel

```
dfNTesA=pd.read_excel("als.xlsx",sheet_name='Teste',index_col=0)
print(dfNTesA.head(2))
```

```
       Matr  T1  T2   T3
Nome
Lalá    133   7   7  NaN
Lelé    131   8   8  8.0
```

als.xlsx




```
dfNTesB=pd.read_excel("als.xlsx",sheet_name='Teste',index_col=1)
print(dfNTesB.head(2))
```

```
       Nome  T1  T2   T3
Matr
133    Lalá   7   7  NaN
131    Lelé   8   8  8.0
```

Métodos Úteis

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

* com **inplace=True**, realiza a operação no DF, não cria uma cópia

`df.sort_values(by,axis=0,ascending=True,na_position='last')` *	Retorna uma cópia do DataFrame ordenado pelos valores **by =** nomes ou lista de nomes para ordenar. Se axis=0 os nomes se referem a colunas. Se axis=1 os nomes se referem a linhas **na_position =** 'last' / 'first', coloca NaN no final/ início
`df.sort_index(axis=0, level=None, ascending=True, sort_remaining=True)` *	Retorna uma cópia do *DataFrame* ordenado pelos labels do índice *sort_remaining* - Se True e o índice é multinível, classifica também pelos outros níveis (em ordem) após classificar pelo nível especificado.
df.rename(mapper=*None*, axis= n/str, index=*None*, columns=*None*) *	Altera o nome do *index* ou *columns.* Retorna um *DataFrame* com o nome da lista de colunas/linhas alterado **Mapper,index,columns -** um dicionário ou função que é aplicado para transformar os valores do eixo. Usar mapper e axis para especificar o eixo ou index/columns **axis =** *n/nome* – eixo a renomear. Padrão: 0 (altera o nome do *index*). Pode ser o nome do eixo `df.rename(columns=str.upper,index={216:2,215:1})`

Métodos Úteis: set_index e reset_index

df.*set_index((keys***, drop=True, append=False, inplace=False, verify_integrity=False)***	Define uma lista/Series/Df como o *index* de um DataFrame **keys -** nome da coluna ou lista de colunas **drop -** a coluna usada para o *index* é descartada, se True **append =-**anexa a coluna ao *index* existente se True, **verify_integrity** -verifica se a nova coluna do *index* está duplicada, se True.
***df.*reset_index(**level=None, drop=False, inplace=False, col_level=0, col_fill=”**) ***	Redefine o *index* para o automático(*0 a n-1)* **level -** int, string ou lista - os seleciona e remove do *index*. **drop –** o *index* é adicionado como coluna de dados, se False **col_level =-**seleciona o nível de *columns a* inserir os rótulos., **col_fill** –objeto para determinar o nome dos outros níveis

pto	x	y	z
p1	1	2	3
p2	4	5	6

```
df.set_index('x')
```

	y	z
x		
1	2	3
4	5	6

```
df.set_index('x',drop=False)
```

	x	y	z
x			
1	1	2	3
4	4	5	6

```
df.set_index('x',drop=False,append=True)
```

		x	y	z
pto	x			
p1	1	1	2	3
p2	4	4	5	6

```
df.set_index('x',append=True,inplace=True)
```

		y	z
pto	x		
p1	1	2	3
p2	4	5	6

pto	x	y	z
p1	1	2	3
p2	4	5	6

```
df.reset_index()
```

	pto	x	y	z
0	p1	2	2	3
1	p2	5	5	6

```
df.reset_index(drop=True)
```

	y	z
0	2	3
1	5	6

Acessar/Alterar ou Incluir/Excluir

Acesso

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

Acesso	Coluna	*df*[*coluna*] **ou** *df.coluna* *df*[*lista de colunas*]	Retorna uma *Series* com os valores da *coluna* ou um *DataFrame* com os elementos da *lista de colunas*
	Linha	*df*.**loc**[*índice*] **ou** *df*.**loc**[*lista de índices*] *df*.**iloc**[*posição*] *df*.**iloc**[*lista de posições*]	Retorna uma Serie com os valores da linha indexada por *índice* ou um *DataFrame* com os elementos da *lista de índices*. **.loc** para os índices criados, **.iloc** para a posição no índice
	Elemento ou Dice	*df*.**loc**[índice][*coluna*] **ou** *df*.**loc**[lista de índs][*lista de cols*]* *df*[*coluna*].**loc**[índice] **ou** *df*[*lista de cols*].**loc**[lista de índs]*	Retorna o valor do elemento indexado por *índice, coluna* ou uma nova Series com os elementos da *lista de índices/colunas*. * Para posição no índice deve ser usado **.iloc**

Mãos na Massa

Considere o Dataframe G:

	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0
Lili	135	1.3	4.6	6.5

Qual o resultado das seguintes seleções?

1) `G[['P1','P3']].iloc[2]`

2) `G['Lalá']`

3) `G.T['Lalá']`

4) `G.T[['Lalá','Lili']].iloc[0:2].T`

1)
```
P1    1.3
P3    6.5
Name: Lili, dtype: float64
```
Series

2)
```
KeyError: 'Lalá'
```

3)
```
Matr    133
P1      6.2
P2      6.9
P3      9.2
Name: Lalá, dtype: object
```
Series

4)

	Matr	P1
Lalá	133	6.2
Lili	135	1.3

DataFrame

Incluir ou Alterar

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

Inc-Alt	Coluna	*df*[*coluna*] **ou** *df.coluna* = **valor** *df*[*lista de colunas*] = **valor ou *Series***	Altera o valor/valores do(s) elemento(s) indexado(s) por *coluna/lista de colunas.* Se a coluna não existe, é incluída.
	Linha	*df*.**loc**[*índice*] =**valor** *df*.**loc**[*lista de índices*] = **valor ou *Series*** *df*.**iloc**[*posição*] =**valor** *df*.**iloc**[*lista de posições*] = **valor ou *Series***	Altera o valor/valores do elemento(s) indexado(s) por *índice/lista de índices.* Se o índice não existe, é incluído.
Alt	Elemento ou Dice	*df.loc*[*índice,coluna*]=**valor ou** *df*.**loc**[lista de índs,*lista de cols*] = **valor**	Altera o valor/valores do elemento(s) indexado(s) por *índice/lista de índices e coluna/lista de colunas*

Excluir

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

Excluir	Coluna	*df.***drop(***coluna***,axis=1)*** ou *df.***drop(***lista de colunas***,axis=1)***	retorna uma cópia do DataFrame sem a coluna/lcolunas especificadas
	Linha	*df.***drop(***índice***)** ou *df.***drop(***lista de índices***)***	retorna uma cópia do DataFrame sem a linha/linhas especificadas

* com inplace=True, realiza a operação na Series, não cria uma cópia

Descrição e Sumarização

Descrição e Sumarização

Medidas de Tendência Central

Média:

`df.mean()`[1]

```
>>>df.mean()
x    13.333333
y    20.000000
z    16.666667
>>>df.mean(axis=1)
p1    20.0
p2    10.0
p3    20.0
```

Mediana:

`df.median()`[1]

```
>>>df.median()
x    10.0
y    20.0
z    10.0
```

Moda:

`df.mode()`[1]

```
>>>df.mode()
      x   y     z
0  10.0  10  10.0
1   NaN  20   NaN
2   NaN  30   NaN
```

df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

1- Com axis=1, operação por linha
2 – Operação aceita nos atributos index e columns

Descrição e Sumarização

Medidas de Tendência Central

Máximo: `df.max()` [1][2]

Mínimo: `df.min()`[1][2]

Índice 1º Mínimo: `df`.idxmin()

Índice 1º Máximo: `df`.idxmax()

```
>>>df.max(axis=1)
p1    30
p2    10
p3    30
dtype: int64
>>>df.min()
x    10
y    10
z    10
dtype: int64
```

Quantil:

`df.quantile (q=%)` [1]

padrão q=0.5

```
>>>df.quantile(axis=1)
p1    20.0
p2    10.0
p3    20.0
```

df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

1- Com axis=1, operação por linha

2 – Operação aceita nos atributos index e columns

Descrição e Sumarização

Medidas de Dispersão

Variância:
`df.var()` [1]
	`>>>df.var()`
	`x  33.333333`
	`y  100.000000`
	`z  133.333333`
	`dtype: float64`

Desvio Padrão:
`df.std()` [1]
	`>>>df.std()`
	`x  5.773503`
	`y  10.000000`
	`z  11.547005`
	`dtype: float64`

df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

1- Com axis=1, operação por linha
2 – Operação aceita nos atributos index e columns