DataFrames

DataFrames do Pandas

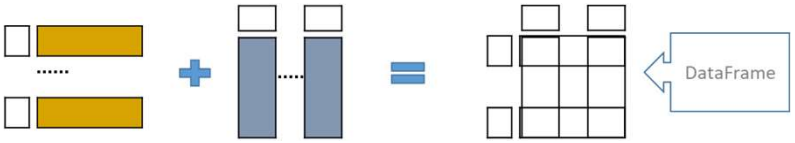
DataFrames: estrutura bidimensional indexada que armazena valores de qualquer tipo.

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

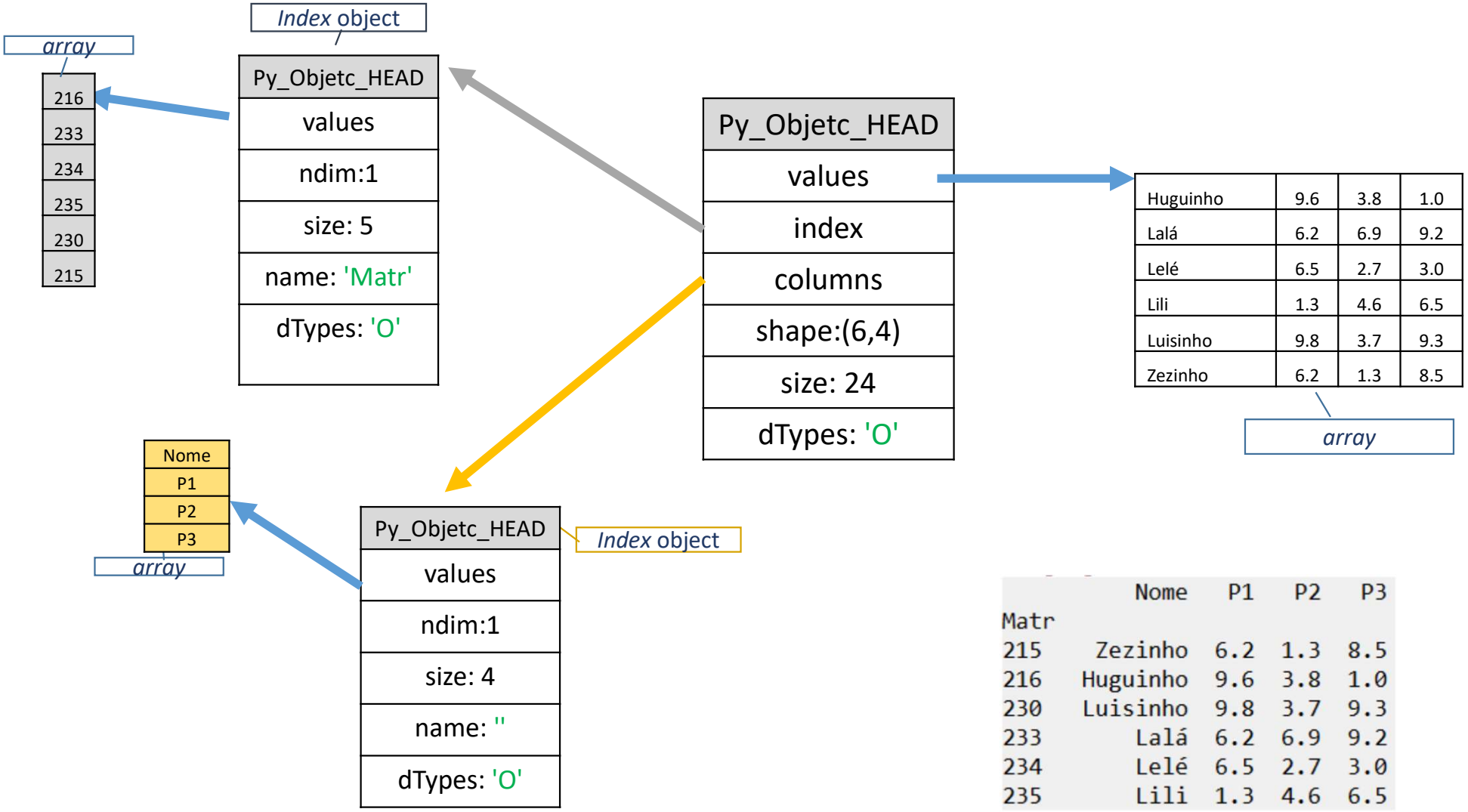


Estrutura tabular com linhas e colunas, similar a uma planilha, composta por:

- **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

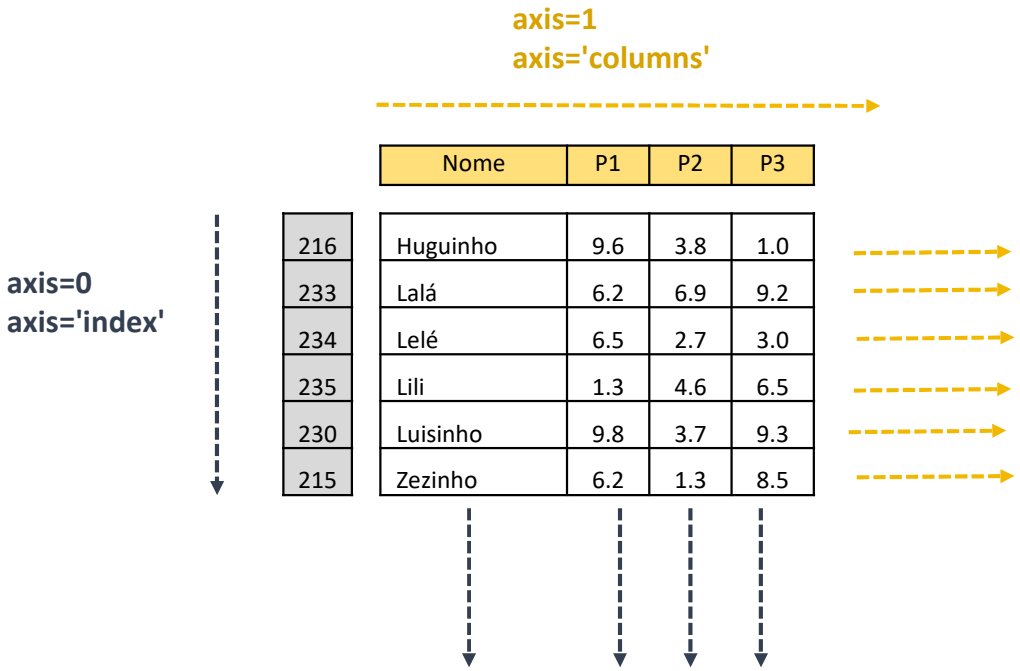
Atributos e Exibição

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"*



Visualização

Traçando gráficos com Pandas

Sintaxe:

```
df.plot(kind='line', xlim=(li,ls), ylim=(li,ls), grid=False,title=None,
                    subplot=False, figsize=(x,y),… )
```

kind= 'line' : linha (default) 'bar' : barra vertical 'barh' : barra horizontal
'hist' : histograma 'box' : boxplot 'scatter' : dispersão 'pie' : pizza
entre outros

[x|y]lim = limites do eixo x ou do eixo y

grid = `True/False,` mostrar grade de linhas

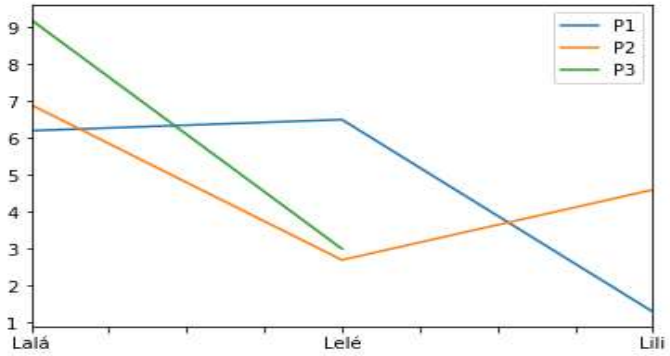
title = título do gráfico

subplots = `True/False`, criar gráficos separados para cada coluna

figsize = *(altura,largura)* em polegadas

dfE:	P1	P2	P3
Lalá	6.2	6.9	9.2
Lelé	6.5	2.7	3.0
Lili	1.3	4.6	NaN

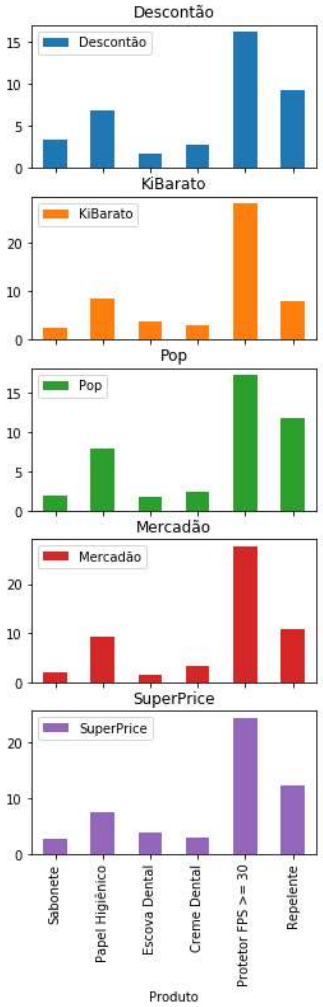
```
dfE.plot()
```



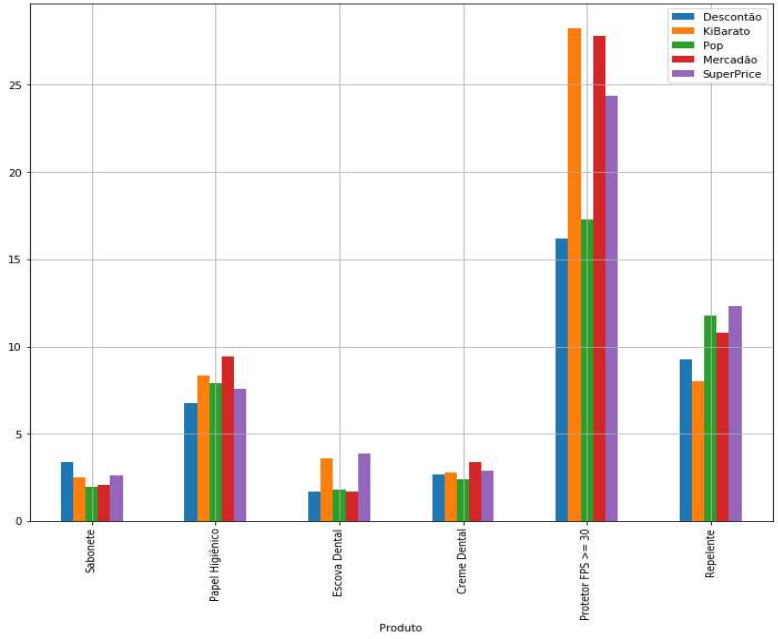
- Valores do eixo x: índices
- Valores do eixo y: colunas

Traçando gráficos com Pandas: Exemplos

```
dfPrHigMerc.plot(
  kind='bar',
  figsize=(4,12),
  subplots=True,
  legend=True
  )
```




```
dfPrHigMerc.plot(
  kind='bar',
  figsize=(4,12),
  grid=True)
```



Valores Ausentes

Excluindo Valores Ausentes

Sintaxe:

```
df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False) *
```

axis: 0/1 . 0 - elimina as linhas com dados ausentes; 1 - remove colunas com dados ausentes;
how: 'any' ou 'all'.

- any: remove as linhas com pelo menos uma coluna NaN e vice-versa
- all: só remove as linhas em que todas as colunas são NaN e vice-versa

thresh: inteiro, número mínimo de NaN para considerar eliminar
subset: na remoçãode linhas, indica o array de colunas onde o método será aplicado;
inplace: caso seja True aplica as alterações no dataset de forma automática;

dfNtd:	Nome	P1	P2	P3
Matr				
133	Lalá	6.2	6.9	9.2
131	NaN	6.5	2.7	3.0
135	Lili	1.3	NaN	6.5

```
>>>dfNtd.dropna()
        Nome   P1   P2   P3
Matr
133     Lalá  6.2  6.9  9.2

>>>dfNtd.dropna(how='all')
         Nome   P1   P2   P3
Matr
133      Lalá  6.2  6.9  9.2
131      NaN   6.5  2.7  3.0
135      Lili  1.3  NaN  6.5
```

DataFrame: fillna

Sintaxe:

```
df.fillna(value=None, axis=None, ascending=True)*
```

Retorna uma cópia do DataFrame substituindo valores NaN

value = scalar, dict, Series, ou DataFrame

Dicionário/ Series / DataFrame de valores especificando qual valor usar para cada índice (para uma Série) ou coluna (para um DataFrame). (os valores que não estão no dict / Series / DataFrame não serão preenchidos).

axis = 0 ou 1

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

dfNt:

Matr	Nome	P1	P2	P3
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0
135	Lili	1.3	4.6	5.0
134	Lolo	NaN	NaN	6.0

```
>>>dfFill=dfNt.fillna(0)
>>>dfFill
```

Matr	Nome	P1	P2	P3
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0
135	Lili	1.3	4.6	5.0
134	Lolo	0.0	0.0	6.0

```
>>>dfF=dfNt.fillna({'P1':0,'P2':2})
>>>dfF
```

Matr	Nome	P1	P2	P3
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0
135	Lili	1.3	4.6	5.0
134	Lolo	0.0	2.0	6.0

Unindo DataFrames

.concat/.append/.merge/.join

https://pandas.pydata.org/pandas-docs/stable/merging.html

Método .concat()

A aplicação do método **.concat** sobre DataFrames, pode realizar tanto a união como a interseção sobre um dos eixos (linha/coluna)

pandas.concat: concatena uma lista ou dicionário de objetos de tipos similares de acordo com o que for estabelecido pelos parâmetros a ser feito com os outros índices

A concatenação é realizada por um dos eixos e as seguintes formas de lidar com o outro eixo estão disponíveis:

a) join='outer': é o padrão e cria a união, classificada, de todos os itens

b) join='inner': cria a interseção de todos os itens

c) join_axes = índice: usa o índice especificado

União - Eixo coluna (axis=0)

Sintaxe:

```
pd.concat([df1,…dfn])
```

```
ldf = [df1, df2, df3]
result = pd.concat(ldf)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

União - Eixo linha (axis=1)

Sintaxe:

```
pd.concat([df1,…dfn],axis=1)
```

```
ldf = [df1, df4]
result = pd.concat(ldf,axis=1)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Os índices das linhas são unidos e ordenados

result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

União - Eixo coluna com índices (keys=[...])

Sintaxe:

```
pd.concat([df1,…dfn],keys=[vl1,…,vln])   ou
pd.concat({chv1:df1,…,chvn:dfn})
```

```
ldf = [df1, df2, df3]
result1 = pd.concat(ldf,keys=['x','y','z'])
result2 = pd.concat({'x':df1,'y':df2,'z':df3})
print(result1.loc['z'])
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

result1

		A	B	C	D
x	0	A0	B0	C0	D0
x	1	A1	B1	C1	D1
x	2	A2	B2	C2	D2
x	3	A3	B3	C3	D3
y	4	A4	B4	C4	D4
y	5	A5	B5	C5	D5
y	6	A6	B6	C6	D6
y	7	A7	B7	C7	D7
z	8	A8	B8	C8	D8
z	9	A9	B9	C9	D9
z	10	A10	B10	C10	D10
z	11	A11	B11	C11	D11

result2

		A	B	C	D
x	0	A0	B0	C0	D0
x	1	A1	B1	C1	D1
x	2	A2	B2	C2	D2
x	3	A3	B3	C3	D3
y	4	A4	B4	C4	D4
y	5	A5	B5	C5	D5
y	6	A6	B6	C6	D6
y	7	A7	B7	C7	D7
z	8	A8	B8	C8	D8
z	9	A9	B9	C9	D9
z	10	A10	B10	C10	D10
z	11	A11	B11	C11	D11

União - Eixo linha com índices (axis=1, keys=[...])

Sintaxe:

```
pd.concat([df1,…dfn],keys=[vl1,…vln],axis=1)   ou
pd.concat({chv1:df1,…,chvn:dfn},axis=1)
```

```
ldf = [df1, df4]
result1 = pd.concat(ldf,keys=['x','y'],axis=1)
result2 = pd.concat({'x':df1,'y':df4},axis=1)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

result1

	x	x	x	x	y	y	y
	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

result2

	x	x	x	x	y	y	y
	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

Interseção - Eixo coluna (join='inner')

Sintaxe:

```
pd.concat([df1,…dfn],join='inner')
```

```
ldf = [df1, df4]
result = pd.concat(ldf,join='inner')
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

result

	B	D
0	B0	D0
1	B1	D1
2	B2	D2
3	B3	D3
2	B2	D2
3	B3	D3
6	B6	D6
7	B7	D7

Interseção - Eixo linha (axis=1, join='inner')

Sintaxe:

```
pd.concat([df1,…dfn],axis=1, join='inner'))
```

```
ldf = [df1, df4]
result = pd.concat(ldf,axis=1,join='inner')
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

result1

	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

Pelo índice de um DF - Eixo Coluna (join_axes=*df.indice*)

Sintaxe:

```
pd.concat([df1,…dfn],join_axes=[df.index/columns])
```

```
ldf = [df1, df4]
result = pd.concat(ldf,join_axes=[df1.columns])
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
2	NaN	B2	NaN	D2
3	NaN	B3	NaN	D3
6	NaN	B6	NaN	D6
7	NaN	B7	NaN	D7

Unir Series como Colunas de DF

Sintaxe:

```
pd.concat([df,s1,…,sn],axis=1,ignore_index=True)
```

As *Series* são transformadas em *DataFrames* com o nome da Series como nome da coluna, se a opção ignore_index=True não estiver presente

```
s1 = pd.Series(['X0', 'X1', 'X2', 'X3'], name='X')
s2 = pd.Series(['_0', '_1', '_2', '_3'])
result1 = pd.concat([df1, s1], axis=1)
result2 = pd.concat([df1,s2,s2],axis=1,ignore_index=True)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2

s1

	X
0	X0
1	X1
2	X2

s2

0	_0
1	_1
2	_2

result1

	A	B	C	D	X
0	A0	B0	C0	D0	X0
1	A1	B1	C1	D1	X1
2	A2	B2	C2	D2	X2

result2

	0	1	2	3	4	5
0	A0	B0	C0	D0	_0	_0
1	A1	B1	C1	D1	_1	_1
2	A2	B2	C2	D2	_2	_2

Pelo índice de um DF - Eixo Linha (axis=1 , join_axis=*df.indice*)

Sintaxe:

```
pd.concat([df1,…dfn],axis=1,join_axes=[df.index/columns])
```

```
ldf = [df1, df4]
result = pd.concat(ldf,axis=1,join_axes=df1.index)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

Mãos na Massa: Unindo DataFrames

I. Construir um DataFrame com os produtos de Higiene que estão na planilha ProdHigieneMerc e com os produtos de Limpeza que estão na planilha ProdLimpezaMerc, ambas no arquivo PrecosProdutosSuperMercados.xlsx

Planilha ProdHigieneMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Planilha ProdLimpezMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

Operações com DataFrames

Operações Aritméticas

Sintaxe: `df.metodoOperação(obj, fill_value=valor)`

obj pode ser um *DataFrame* ou uma *Series* ou um escalar. Os dados são alinhados pelas colunas e pelos índices. Retorna um DataFrame com a união das colunas e dos *labels* das linhas. Se o argumento fill_value está presente e não há sobreposição nos índices, utiliza *valor* para o cálculo, senão o valor é NaN

```
df1.add(df2,fill_value=0)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

+

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

=

dfR	a	b	c	d
A	11	22	33	100
B	44	55	66	200
C	77	88	99	300

```
df1.sub(df2,fill_value=0)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

-

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

=

dfR	a	b	c	d
A	-9	-18	-27	-100
B	-36	-45	-54	-200
C	-63	-72	-81	-300

```
df1.mul(df2)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

*

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

=

dfR	a	b	c	d
A	10	40	90	NaN
B	160	250	360	NaN
C	490	650	810	NaN

Operações Aritméticas

```
df1.div(df2,fill_value=1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

/

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

=

dfR	a	b	c	d
A	0.1	0.1	0.1	0.01
B	0.1	0.1	0.1	0.005
C	0.1	0.1	0.1	0.003

```
df1.floordiv(df2,fill_value=1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

//

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

=

dfR	a	b	c	d
A	0	0	0	0.0
B	0	0	0	0.0
C	0	0	0	0.0

```
df1.mod(df2,fill_value=0)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

%

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

=

dfR	a	b	c	d
A	1	2	3	0.0
B	4	5	6	0.0
C	7	8	9	0.0

Métodos Aritméticos DF e Series

```
df1.add(s1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

+

s1	
a	10
b	20
c	30
d	40

=

dfR	a	b	c	c
A	11	22	33	NaN
B	14	25	36	NaN
C	17	28	39	NaN

```
df1.sub(s1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

-

s1	
a	10
b	20
c	30
d	40

=

dfR	a	b	c	c
A	-9	18	-27	NaN
B	-6	-15	-24	NaN
C	-3	-12	-21	NaN

```
df1.mul(s1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

*

s1	
a	10
b	20
c	30
d	40

=

dfR	a	b	c	c
A	10	40	90	NaN
B	40	100	180	NaN
C	70	160	270	NaN

Métodos Aritméticos DF e Series

```
df1.div(s1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

/

s1	
a	10
b	20
c	30
d	40

=

dfR	a	b	c	c
A	0.1	0.10	0.1	NaN
B	0.4	0.25	0.2	NaN
C	0.7	0.40	0.3	NaN

```
df1.floordiv(s1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

//

s1	
a	10
b	20
c	30
d	40

=

dfR	a	b	c	c
A	0	0	0	NaN
B	0	0	0	NaN
C	0	0	0	NaN

```
df1.mod(s1)
```

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

%

s1	
a	10
b	20
c	30
d	40

=

dfR	a	b	c	c
A	1	2	3	NaN
B	4	5	6	NaN
C	7	8	9	NaN

Solução: Preço total do Kit em cada supermercado

I. Exibir o preço a pagar por cada produto do kit higiene em cada um dos supermercados, considerando a quantidade determinada na cesta básica

Planilha ProdHigieneMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Planilha Produtos

Produto	Categoria	Unidade	Quantidade
Sabonete	Higiene	90 g	3
Papel Higiênico	Higiene	Pct 4 unidades	2
Escova Dental	Higiene	Unidade	2
Creme Dental	Higiene	90 g	1
Protetor Solar FPS >= 30	Higiene	100 ml	1
Repelente de insetos	Higiene	300 ml	1
Detergente Liquido	Limpeza	500 ml	1
Lã de Aço	Limpeza	Pct 4 unidades	1
Sabão em Pó	Limpeza	500 g	1
Desinfetante	Limpeza	1000 ml	1

Produto	Qt	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Tot Descontão	Tot KiBarato	Tot Pop	Tot Mercadão	Tot SuperPrice
Sabonete	3	3.39	2.48	1.97	2.09	2.6	10.17	7.44	5.91	6.27	7.8
Papel Higiênico	2	6.75	8.36	7.92	9.43	7.57	13.5	16.72	15.84	18.86	15.14
Escova Dental	2	1.69	3.58	1.8	1.67	3.88	3.38	7.16	3.6	3.34	7.76
Creme Dental	1	2.69	2.8	2.37	3.35	2.86	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	1	16.21	28.23	17.28	27.8	24.37	16.21	28.23	17.28	27.8	24.37
Repelente	1	9.24	8.02	11.76	10.81	12.33	9.24	8.02	11.76	10.81	12.33

Desenvolvendo a Solução

i. Selecionar a coluna quantidade dos produtos de Higiene

Planilha Produtos

Produto	Categoria	Unidade	Quantidade
Sabonete	Higiene	90 g	3
Papel Higiênico	Higiene	Pct 4 unidades	2
Escova Dental	Higiene	Unidade	2
Creme Dental	Higiene	90 g	1
Protetor Solar FPS >= 30	Higiene	100 ml	1
Repelente de insetos	Higiene	300 ml	1
Detergente Liquido	Limpeza	500 ml	1
Lã de Aço	Limpeza	Pct 4 unidades	1
Sabão em Pó	Limpeza	500 g	1
Desinfetante	Limpeza	1000 ml	1

Series Qt

Produto	Qt
Sabonete	3
Papel Higiênico	2
Escova Dental	2
Creme Dental	1
Protetor FPS >= 30	1
Repelente	1

a) Filtrar linhas da Categoria == 'Higiene'
b) Selecionar a coluna quantidade (gera uma Series)
c) Renomear a Series de 'Quantidade' para 'Qt'

Método Úteis

Outros Métodos para União, Substituição e Atualização de DataFrame:

Sintaxe:

`df.append(df)` – Cria uma cópia com os elementos do DataFrame recebido incluídos no final, alinhados pelo columns

`df.replace(to_replace=valor,value=novo)` – Cria uma cópia, substituindo todas as ocorrências de valor por novo

`df.update(df)` – Altera atuais valores pelos valores recebidos, alinhando pelo índice

Mãos na Massa

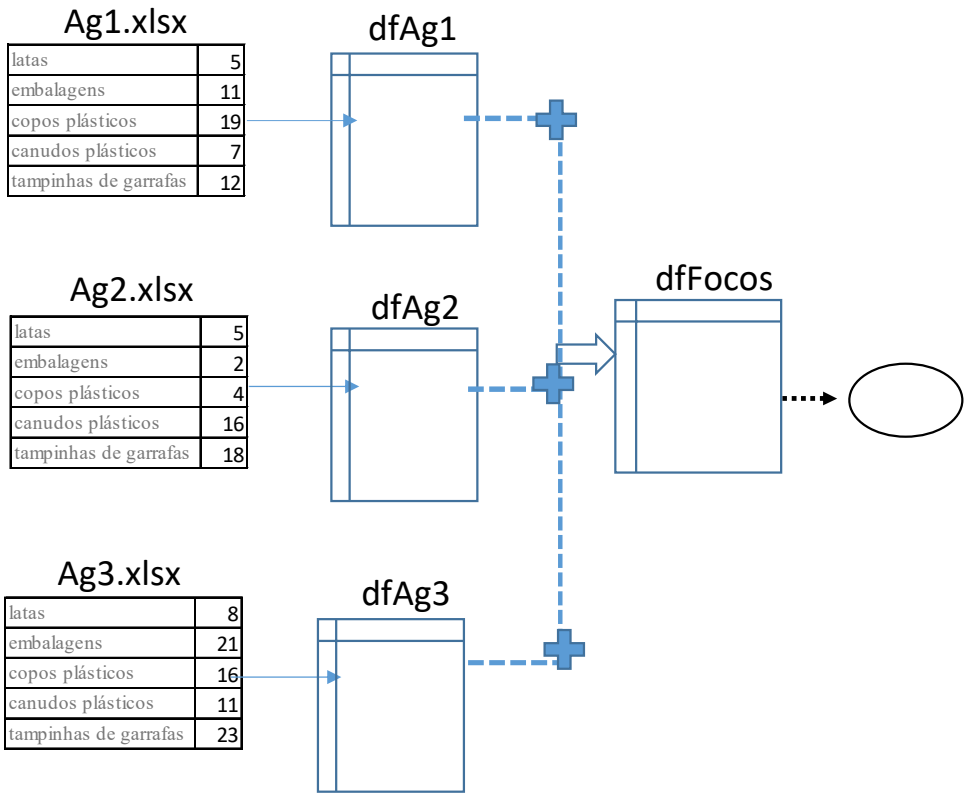
Diariamente, 3 agentes da Vigilância Sanitária visitam uma região para detectar focos de larvas do mosquito Aedes aegypti nos seguintes locais:

latas, embalagens, copos e canudos plásticos, garrafas, tampinhas de garrafas, vasos de plantas, jarros de flores, bromélias, caixas d´água, tambores, latões, cisternas, calhas, piscinas, vasos sanitários, pneus velhos, sacos plásticos, lixeiras, bueiros, ralos, lonas e lajes.

A quantidade de focos encontrados por localização são registradas em arquivos Excel denominados Ag_n onde *n* varia de 1 a 3. Caso não seja encontrado focos em alguma localização, ela não consta no arquivo.

Construa um script que mostre para cada agente, a quantidade de focos encontrada por tipo de localização, a localização com maior incidência, quantidade total e a visualização gráfica das quantidades por localização.

Análise da Solução por Agente



I. Criar DF de Series:

Criar DF com o arq $Ag_i.xlsx$

Concatenar DFs de Ags

II. Tratar DF Todos:

Exibe DF

Sumariza DF

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Uma Solução: Focos por Agente

```
import pandas as pd

def geraNome(n):
    return "AG"+str(n+1)

def montaDataFrame(qt):
    #Cria o nome do arquivo do agente
    ldfs=[]
    for i in range(0,qt):
        #Monta o nome do arquivo do agente
        arq=geraNome(i)+".xlsx"
        #Cria um df a partir do arquivo, renomeando a coluna
        df= df=pd.read_excel(arq,header=None,index_col=0)
        df.rename(columns={1:'Ag'+str(i+1)},inplace=True)
        #Adiciona o DF Transposto à lista
        ldfs.append(df.T)
    dfFocos=pd.concat(dfAgs)
    #Como há focos inexistentes em alguns agentes, substitui NaN por 0
    dfFocos.fillna(value=0,inplace=True)
    return dfFocos
```

Uma Solução: Focos por Agente

```
def trataAg(df):
    #Sumarização por agente
    total=df.sum(axis=1)
    maior=df.max(axis=1)
    local=df.idxmax(axis=1)
    df['Total de Focos']=total
    df['Maior Incidência']=maior
    df['Local Maior Incidência']=local
   return

#Cria o DF com dados dos agentes
dfFocos= montaDataFrame(3)
print(dfFocos)
#Sumarização por agente
trataAg(dfFocos):
```

Análise da Solução: Totais Gerais

Como saber:

	Ag1	Ag2	Ag3
latas	125	13	1
embalagens	11	8	2
copos plásticos	4	18	22
canudos plásticos	10	23	15
...	...	...	...
lajes	21	6	3

I. O total de focos encontrados na região por localização?
   somar as Linhas

II. Qual a localização com mais focos na região?
   Encontrar o maior valor e o local da soma das linhas

III. A distribuição gráfica dos focos por localização na região?
   Construir um gráfico de barras com o total de focos encontrados por localização

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Uma Solução

```
def resumoTotais(df):
    total=df.sum(axis=1)
    maior=total.max()
    local=total.idxmax()
    dfTot=pd.DataFrame(total, columns=['Total'])
    dfTot.index.name='Local'
    print(dfTot)
    print('\n\nLocal de Maior Localização: {} com {:.2f} focos' .format(local,maior))

    return

#Cria o DF com dados dos agentes
dfFocos= montaDataFrame(3)
print(dfFocos)
#Sumarização por agente
trataAg(dfFocos):
#Sumarização geral
resumoTotais(dfFocos):
print(dfFocos
```