Python para usuárias de Excel

Organizadora:
Deborah Foroni

Compartilhar

O que é PyLadies?

Focado em aumentar a atividade e a liderança das **mulheres** na comunidade **Python**

Missão Global : **promover, educar e desenvolver uma comunidade de Python plural** por meio de divulgação, educação, conferências, eventos e reuniões sociais.

PyLadies São Paulo

O PyLadies São Paulo é um capítulo das PyLadies internacional cuja missão é incentivar quaisquer mulheres a aprenderem Python, ensinarem e motivarem outras a fazerem o mesmo.

Nosso lema:

“O pouco que você sabe pode ser muito para quem não sabe nada!”

Agenda do Dia

Notebooks

Tipos de dados

Estruturas dos dados

Bibliotecas

Pandas

pyladies São Paulo

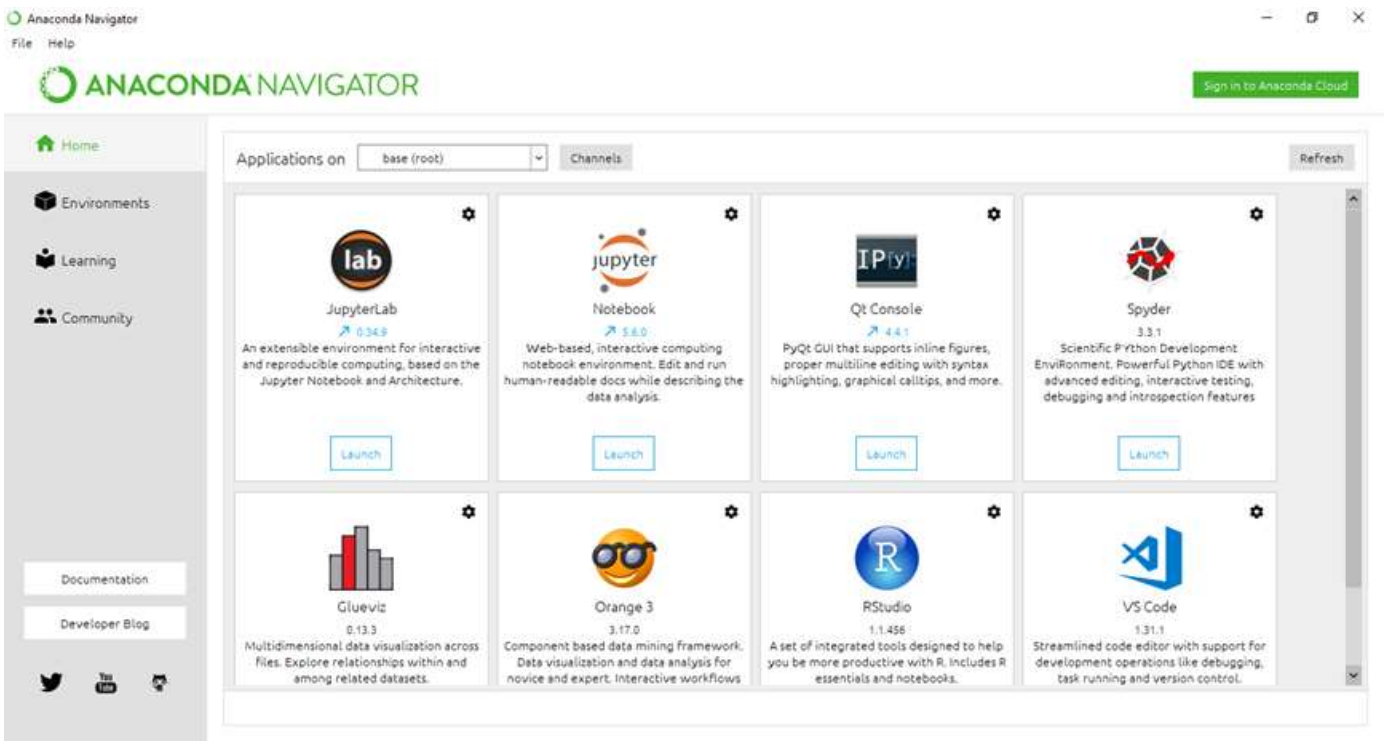
Jupyter Notebook

Nesta etapa, vamos juntas aprender como usar o ambiente baseado no Jupyter Notebook.

pyladies São Paulo

Anaconda

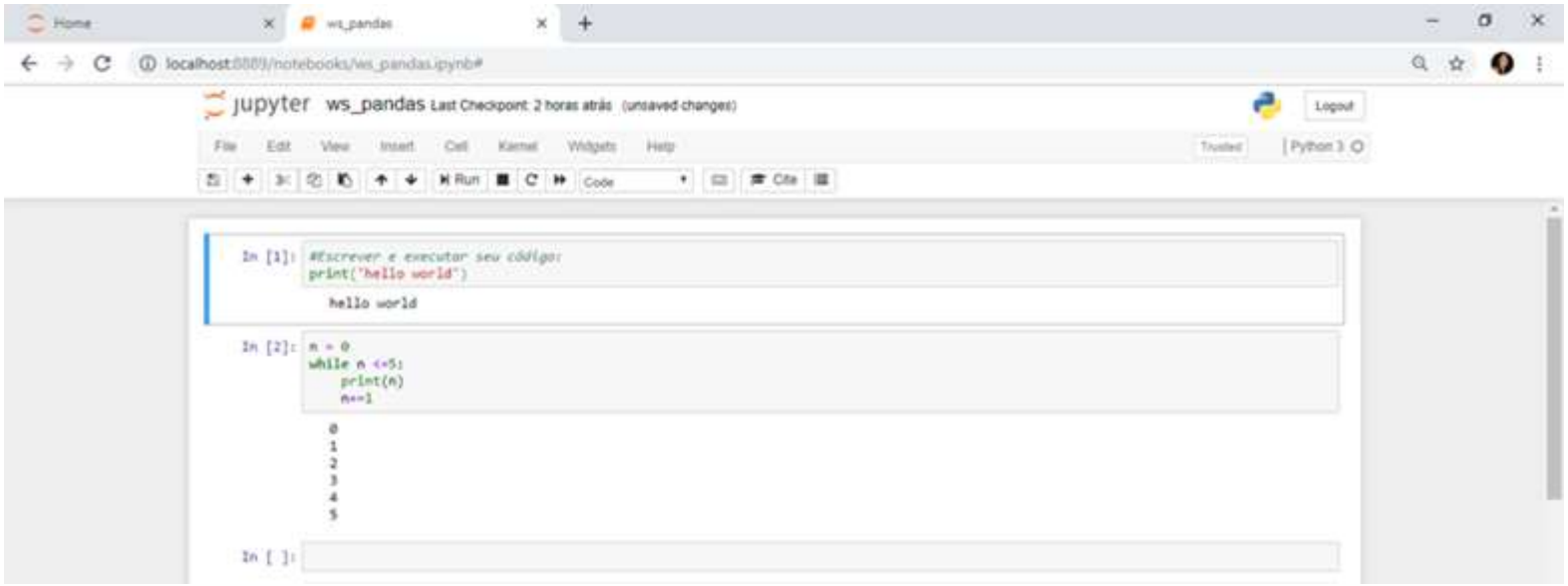
Distribuição gratuita de código aberto das linguagens Python e R. Permite gerenciar pacotes via interface gráfica e linha de comando (conda prompt).



pyladies São Paulo

Jupyter Notebook

Notebooks são documentos que contém códigos, textos, gráficos, links, equações etc. e por conta da quantidade de recursos são muito utilizados em análises.



Jupyter Notebook

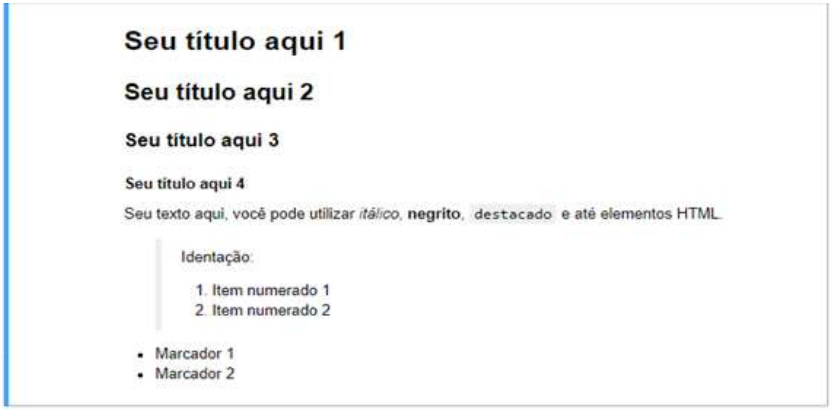
As células configuradas como “Markdown” permite escrever textos simples e formatados.

Modo edição

```
# Seu título aqui 1
## Seu título aqui 2
### Seu título aqui 3
#### Seu título aqui 4
Seu texto aqui, você pode utilizar _itálico_, **negrito**, `destacado` e até elementos HTML.
> Identação:
1. Item numerado 1
2. Item numerado 2

- Marcador 1
- Marcador 2
```

Célula executada

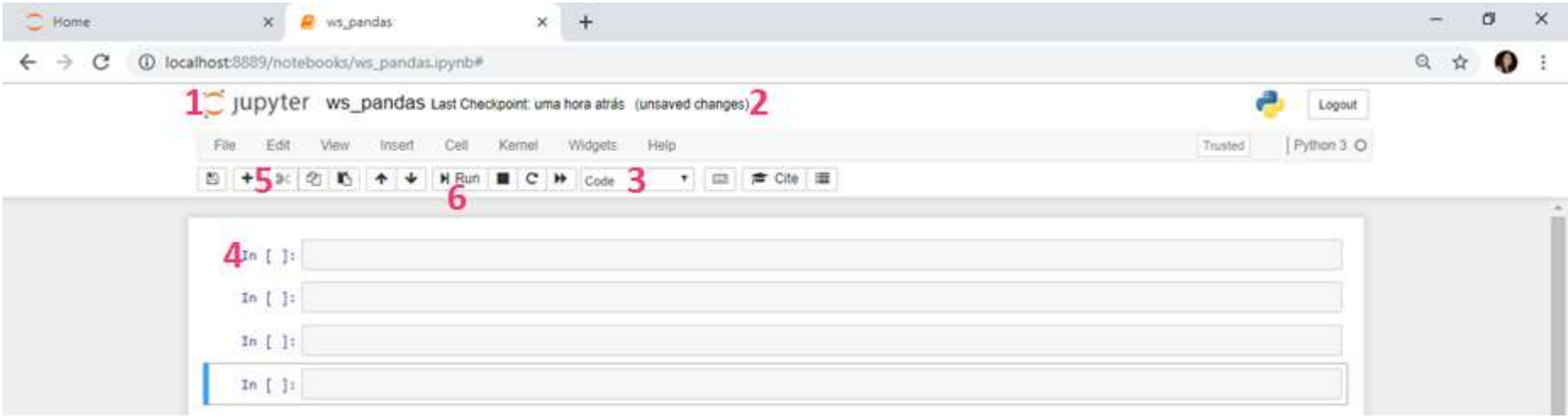



```
<h3> Markdown com HTML </h3>

<p>Estou criando um link para
<a href="https://github.com/angelica93/GEDS">Grupo de estudos Data Science</a>.
</p>
```

pyladies São Paulo

Jupyter Notebook



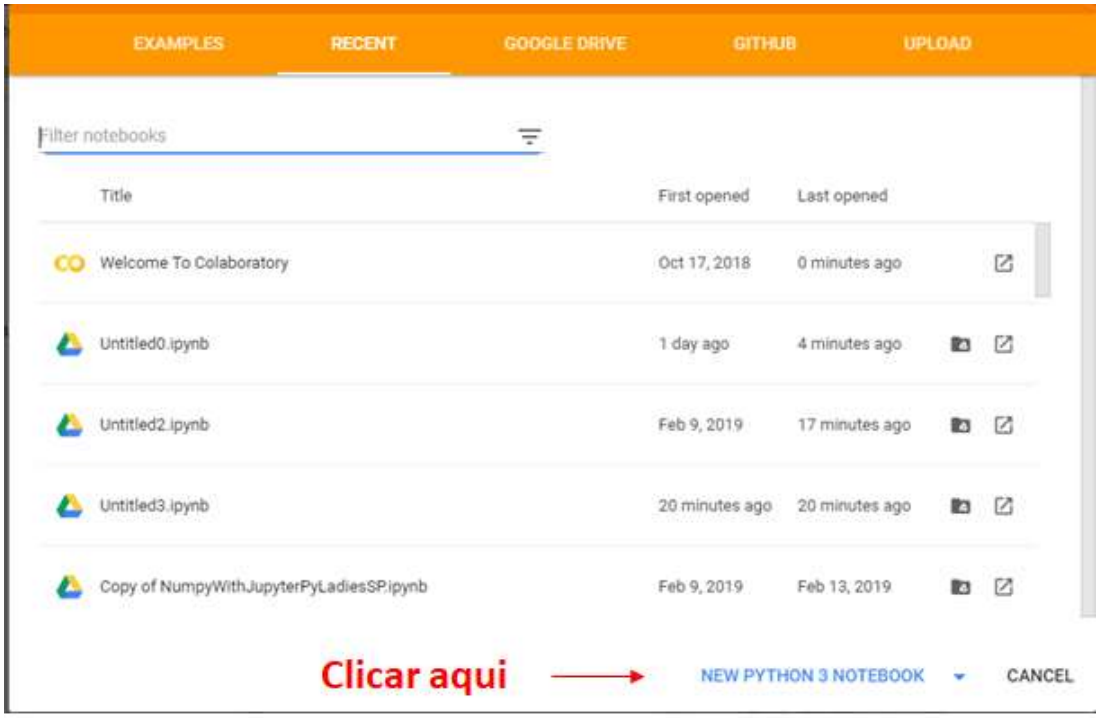
1. Visualização dos arquivos do diretório atual;
2. Nome do arquivo e quando foi salvo;
3. Configuração da célula, pode ser: code, markdown ou Raw NBConvert;
4. Célula para digitar texto ou código;
5. Adicionar célula abaixo; e
6. Executar a célula atual.

pyladies São Paulo

Google Colaboratory

O Colab segue o mesmo padrão do Jupyter Notebook. Nele é possível adicionar células de código, texto, importar arquivos etc. Para ter acesso ao Google Colab, basta logar na sua conta Google e acessar o link https://colab.research.google.com;

Aparecerá uma tela como



pyladies São Paulo

Notebooks

São ambientes online que conseguimos criar códigos organizados.

Vamos abrir agora o Colab ou Jupyter!

pyladies
São Paulo

Tipos de Dados

Nesta etapa, vamos juntas aprender os tipos de dados.

pyladies São Paulo

Tipos de dados

Quantitativos - Quantifica ou mede

Discretos: Assumem valores em um conjunto especificado de números.

Contínuos: Assumem valores em um intervalo contínuo de números.

Qualitativos - Característica ou qualidade

Nominal: Característica que não possui ordem.

Ordinal: Característica que possui uma ordem de grandeza.

pyladies São Paulo

Tipos de Dados

Quantitativos

Discretos:
- Quantidade de pessoas na sala?
- Quantidade de dias no mês?

Contínuos:
- Qual sua altura?
- Qual a distância daqui até sua casa?

Qualitativos

Nominal:
- Qual seu Estado? (SP, MG, RJ, Outros)
- Qual gênero você se identifica?

Ordinal:
- Avaliação curso (Ruim a Ótimo).
- Qual seu nível de escolaridade?

pyladies São Paulo

Tipos de Dados - Exemplo PyLadies

ID	Estado Origem	Idade	Escolaridade	Trabalha como Programadora	Renda Mensal
1	SP	36	4	S	3737,52
2	SP	25	2	N	400,00
3	MG	34	3	S	2366,14
4	RJ	23	3	S	2841,29
5	SP	31	4	N	800
6	SP	34	5	S	3433,02
7	SP	39	5	S	2752,74
8	PE	24	3	S	3682,33
9	RJ	29	3	S	2359,28
10	SP	27	3	S	2119,15
11	SP	30	3	S	3326,79
12	SP	25	4	S	2684,05
13	SP	23	2	S	3507,84
14	SP	16	1	N	0
15	SP	36	4	N	800

Legenda Escolaridade

1	Ensino Medio Completo
2	Graduanda
3	Graduação Completa
4	Pós graduanda
5	Pós graduação completa

*Dados meramente ilustrativos.

pyladies São Paulo

Tipos de Dados - Dados Qualitativos

ID	Estado Origem	Idade	Escolaridade	Trabalha como Programadora	Renda Mensal
1	SP	36	4	S	3737,52
2	SP	25	2	N	400,00
3	MG			S	2366,14
4	RJ			S	2841,29
5	SP			N	800
6	SP	34	5	S	3433,02
7	SP	39	5	S	2752,74
8	PE	24	3	S	3682,33
9	RJ	29	3	S	2359,28
10	SP	27	3	S	2119,15
11	SP	30	3	S	3326,79
12	SP	25	4	S	2684,05
13	SP	23	2	S	3507,84
14	SP	16	1	N	0
15	SP	36	4	N	800

Dados Qualitativos

Legenda Escolaridade

1	Ensino Medio Completo
2	Graduanda
3	Graduação Completa
4	Pós graduanda
5	Pós graduação completa

pyladies São Paulo

Tipos de Dados - Categóricos

ID	Estado Origem	Idade	Escolaridade	Trabalha como Programadora	Renda Mensal
1	SP	36	4	S	3737,52
2	SP	25	2	N	400,00
3	MG	34	3	S	2366,14
4	SP	23	4	N	2841,29
5	SP	31	4	N	800
6	SP	34	5	S	3433,02
7	SP	39	5	S	2752,74
8	PE	24	3	S	3682,33
9	RJ	29	3	S	2359,28
10	SP	27	3	S	2119,15
11	SP	30	3	S	3326,79
12	SP	25	4	S	2684,05
13	SP	23	2	S	3507,84
14	SP	16	1	N	0
15	SP	36	4	N	800

Nominal **Ordinal** **Nominal**

Legenda Escolaridade

1	Ensino Medio Completo
2	Graduanda
3	Graduação Completa
4	Pós graduanda
5	Pós graduação completa

pyladies São Paulo

Tipos de Dados - Dados Quantitativos

ID	Estado Origem	Idade	Escolaridade	Trabalha como Programadora	Renda Mensal
1	SP	36	4	S	3737,52
2	SP	25	2	N	400,00
3	MG	34	3	S	2366,14
4	RJ	23			2841,29
5	SP	31			800
6	SP	34	5	S	3433,02
7	SP	39	5	S	2752,74
8	PE	24	3	S	3682,33
9	RJ	29	3	S	2359,28
10	SP	27	3	S	2119,15
11	SP	30	3	S	3326,79
12	SP	25	4	S	2684,05
13	SP	23	2	S	3507,84
14	SP	16	1	N	0
15	SP	36	4	N	800

Dados Quantitativos

Legenda Escolaridade

1	Ensino Medio Completo
2	Graduanda
3	Graduação Completa
4	Pós graduanda
5	Pós graduação completa

pyladies São Paulo

Tipos de Dados - Numéricos

ID	Estado Origem	Idade	Escolaridade	Trabalha como Programadora	Renda Mensal
1	SP	36	4	S	3737,52
2	SP	25	2	N	400,00
3	MG	[illegible]	3	S	[illegible]
4	RJ	[illegible]	3	S	[illegible]
5	SP	31	4	N	800
6	SP	34	5	S	3433,02
7	SP	39	5	S	2752,74
8	PE	24	3	S	3682,33
9	RJ	29	3	S	2359,28
10	SP	27	3	S	2119,15
11	SP	30	3	S	3326,79
12	SP	25	4	S	2684,05
13	SP	23	2	S	3507,84
14	SP	16	1	N	0
15	SP	36	4	N	800

Discreto

Contínuo

Legenda Escolaridade

1	Ensino Medio Completo
2	Graduanda
3	Graduação Completa
4	Pós graduanda
5	Pós graduação completa

pyladies São Paulo

Estrutura de Dados

Nesta etapa, vamos juntas aprender as estrutura de dados.

pyladies São Paulo

Estruturas dos Dados

O que é um dataframe??

DataFrame é uma estrutura de dados bidimensional - parecida com uma tabela de excel ou um banco de dados.

pyladies São Paulo

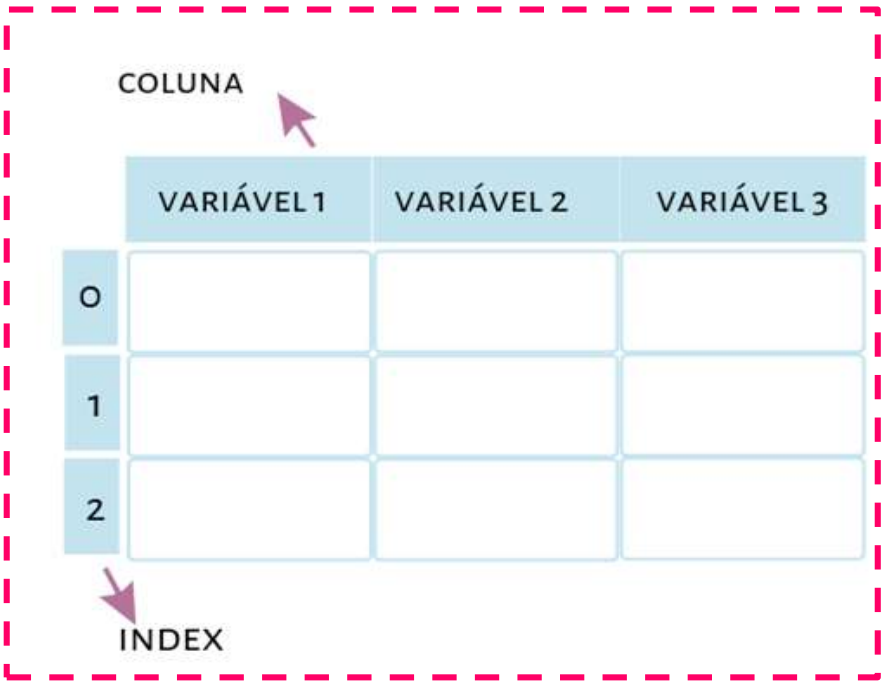
Estruturas dos Dados

Estrutura de dados bidimensional (colunas e linhas) cujo índice começa no **zero.** Lembrando que no Excel o índice começa no **1**.

O *dataframe* contém colunas que armazenam diferentes tipos de informações (*string, float, integer* e *etc*)

Ele é uma classe de objeto da biblioteca **Pandas**.

dataframe



pyladies São Paulo

Anatomia de um dataframe Pandas

	id	age	sex	bmi	children	smoker	region	charges
0	1	19	female	27.90	0	yes	southwest	1.688492e+07
1	2	18	male	33.77	1	no	southeast	1.725552e+07
2	3	28	male	33.00	3	no	southeast	4.449462e+06
3	4	33	male	22705.00	0	no	northwest	2.198447e+09
4	5	32	male	28.88	0	no	northwest	3.866855e+07
...	...	...	...	...	...	...	...	...
1333	1334	50	male	30.97	3	no	northwest	1.060055e+08
1334	1335	18	female	31.92	0	no	northeast	2.205981e+07
1335	1336	18	female	36.85	0	no	southeast	1.629834e+07
1336	1337	21	female	25.80	0	no	southwest	2.007945e+06
1337	1338	61	female	29.07	0	yes	northwest	2.914136e+08

[1338 rows x 8 columns]

pyladies São Paulo

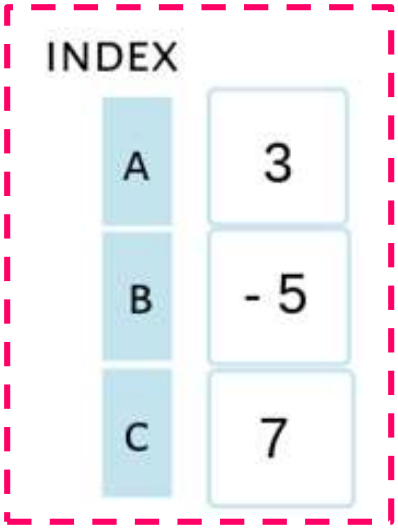
Estruturas dos Dados

E o que são series ??

DataSerie é estrutura unidimensional como uma coluna do *excel*.

pyladies São Paulo

Estrutura dos Dados



Series

Um *array* unidimensional e rotulado capaz de armazenar qualquer tipo de dado.

```
s = pd.Series([3,-5,7,4], index = ['a','b','c','d'])
print(s)
```

```
a    3
b   -5
c    7
d    4
dtype: int64
```

pyladies São Paulo

Algumas definições

pyladies São Paulo

O que é atributo e método?

Atributo:

objeto.atributo
planilha.sheet_names

Nos diz algo e sempre conectado ao um objeto.

Método:

objeto.método()
planilha.parse()

Faz algo para nós e sempre conectado ao um objeto.

pyladies São Paulo

O que é Script?

É uma série de instruções com uma ordem específica que gera no final um resultado.

Como por exemplo: uma receita de Beijinho

1 leite condensado (lata ou caixinha)

1 colher (sopa) de manteiga

4 colheres (sopa) de coco seco ralado

coco seco ralado para passar os docinhos

Script Python

```
import pandas as pd
import numpy as np

dados = pd.read_excel('insurance.xlsx')
dados.head()
```

pyladies São Paulo

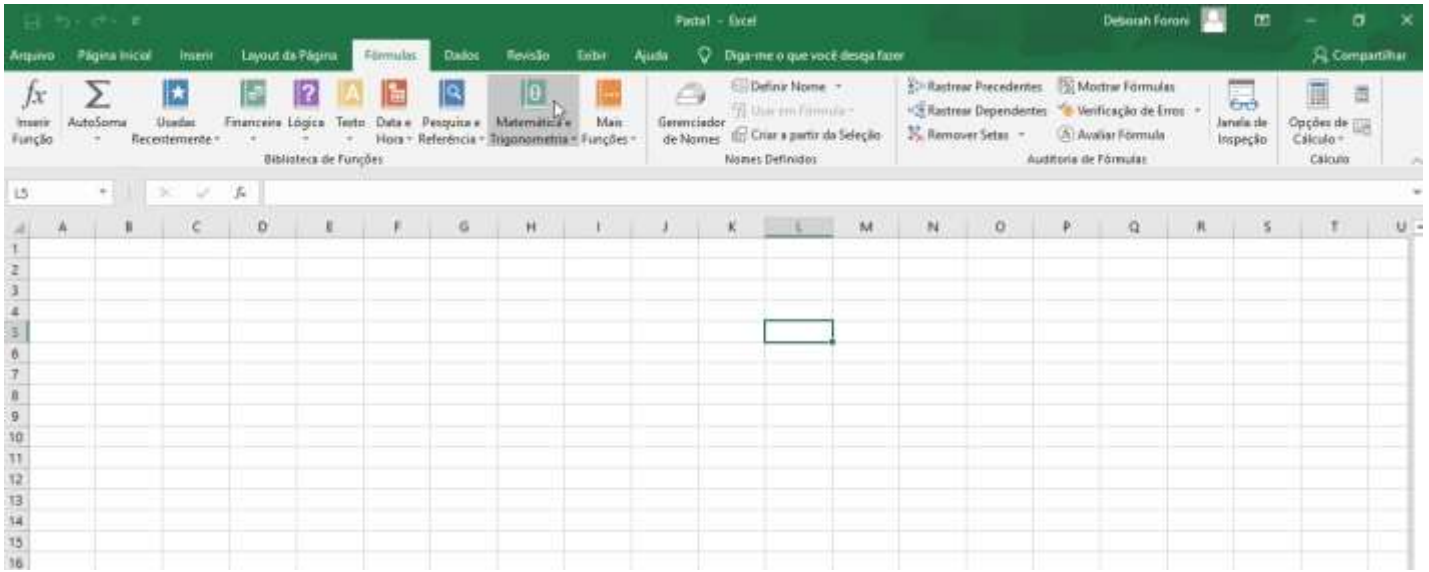
Programação Orientada a Objeto

O Python é uma linguagem orientada a objeto, e possui excelentes bibliotecas.

E o que isso quer dizer?
Que ele tem uma coisa dentro de uma outra coisa.

A animação abaixo irá auxiliar a compreensão, quase sempre usamos a função SOMA do Excel, certo? Mas sabemos que ela está dentro de um pacote chamado **Matemática e Trigonometria**?

Objeto Orientado da planilha



pyladies São Paulo

Programação Orientada a Objeto

Objeto Orientado Python

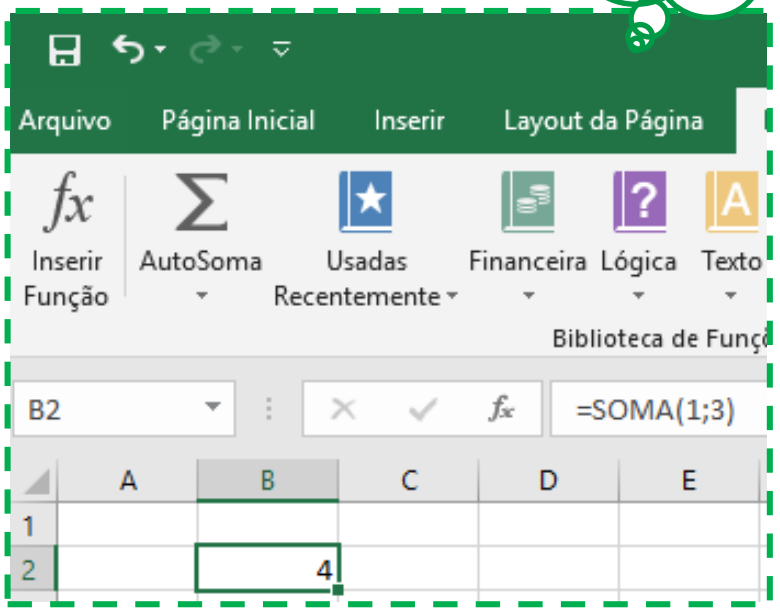
Fazendo um comparativo com o Excel, o Python possui uma biblioteca chamada **Numpy,** que tem um função chamada **SUM** igual a do **Excel.**

Python

```
[11] np.sum([1,1])

     2
```

Excel



pyladies São Paulo

O que é biblioteca no Python?

```
[8]  import numpy as np
```

Numpy é a biblioteca


```
[12] np.sum()
```

Sum é a função que importamos da biblioteca Numpy.

O ponto que deixa acessar a função dentro da biblioteca.

pyladies São Paulo

Biblioteca Pandas

Neste Workshop iremos trabalhar bastante com a biblioteca Pandas:

A biblioteca Pandas têm código aberto que fornece estruturas de dados de alta performance e fácil de utilizar, é também uma ótima ferramenta para manipulação de dados.

O nome Pandas vem da combinação de Panel Data e Python Data Analysis.

Ela possui uma ótima documentação: https://pandas.pydata.org/

Pandas

pyladies São Paulo

Como importar os dados?

Nesta etapa, vamos juntas aprender como importar o arquivo em Excel para o Python.

pyladies São Paulo

Como importar dados para o Python?

A primeira etapa é importar o arquivo para o Python, assim conseguiremos trabalhar os dados.

O método (commando) que iremos usar para a importação será o ***read_excel*** da biblioteca **Pandas**, este método suporta as extensões ***xls*** e ***xlsx***.

Existem outros métodos para importar arquivos de outras extensões (*csv, txt e etc*).

1° Passo

```
import pandas as pd
```

2° Passo

```
from google.colab import files
uploaded = files.upload()
```

```
dados = pd.read_excel('salarios.xlsx')
print(dados)
```

pyladies São Paulo

Agora é com você!

A partir de agora vamos começar a trabalhar com o notebook Colab.

https://colab.research.google.com

```
[2] from google.colab import files
    uploaded = files.upload()
```

Escolher arquivos | salarios.xlsx

- **salarios.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 30407299 bytes, last modified: 13/02/2020 - 100% done

```
Saving salarios.xlsx to salarios.xlsx
```

✓ **Arquivo importado com sucesso.**

```
# Estamos carregando o arquivo Excel e colocando dentro de um objeto, neste caso dentro do objeto "dados".
dados = pd.read_excel('salarios.xlsx')
print(dados)
```

✓ **Arquivo lido com sucesso.**

pyladies São Paulo

Importando outras sheets no Python

Por padrão o **Pandas** importa a primeira *sheet* do arquivo, mas na vida real sabemos que não é assim que funciona e que precisamos também de outras *sheets*, abaixo vamos importar uma *sheet* específica.

Para isso usaremos o argumento chamado *sheet_name*, com ele consegui mos dizer qual a *sheet* que queremos importar.

pd.read_excel('nome_arquivo', sheet_name = 'nome_sheet')

```
new_sheet = pd.read_excel('salarios.xlsx', sheet_name='salario_mensal_02')
print(new_sheet)
```

pyladies São Paulo

Usando uma coluna como índice

As vezes necessitamos usar uma coluna como índice, e para isto usaremos o argumento do ***read_excel*** chamado **index_col**.

pd.read_excel('nome_arquivo', sheet_name = 'nome_sheet', index_col=0)

```
indice = pd.read_excel('Salarios.xlsx', sheet_name='salario_mensal_01', index_col=0)
print(indice)
```

pyladies São Paulo

Pulando linhas e colunas

Por padrão o método ***read_excel*** assume que a primeira linha do *dataframe* é uma lista de colunas formando assim o cabeçario do *dataframe* (os nomes das colunas).

Para isto necessitamos dos argumentos ***header*** e ***skiprows*** para manipulação do cabeçario e das linhas.

pd.read_excel('nome_arquivo', header='true ou false', skiprows='linha')

```
skip = pd.read_excel('Salarios.xlsx', sheet_name='salario_mensal_01', header=None, skiprows=1, index_col=0)
print(skip)
```

pyladies São Paulo

Importando colunas específicas

Muitas vezes não necessitamos de todas as colunas do *dataframe*, e para isto existe um argumento que nos ajuda selecionar apenas as colunas que necessitamos, que é o ***usecols***.

pd.read_excel('nome_arquivo', usecols='nome das colunas')

```
col = pd.read_excel('Salarios.xlsx', sheet_name='salario_mensal_01', header=None, skiprows=1, usecols='A,D')
print(col)
```

pyladies São Paulo

Agora é com você! – Desafio 1

- ✔ Importar o arquivo *csv* chamado:

  ‘monthly_salary_brazil’.

pyladies São Paulo

Resposta - Desafio 1

✔ Importar o arquivo *csv* chamado:

‘monthly_salary_brazil’.

```
# Estamos carregando o arquivo CSV e colocando dentro de um objeto, neste caso dentro do objeto "dados".
dados = pd.read_csv('monthly_salary_brazil.csv', error_bad_lines=False)
print(dados)
```

pyladies São Paulo

Importar múltiplas abas

Nesta etapa, vamos juntas aprender como importar múltiplas abas para o *Python*.

pyladies São Paulo

Biblioteca Google Drive

Primeiro vamos aprender uma outra maneira de importar arquivos no **Colab**.

```
# Biblioteca que faz chamada para o Google Drive
from google.colab import drive

# Isso solicitará autorização com a sua conta do Google Drive
drive.mount('/content/drive')

# Lista o conteúdo do seu Google Drive
!ls "/content/drive/My Drive"
```

pyladies São Paulo

Importando múltiplas abas

Agora vamos usar a função *ExcelFile* para a leitura do arquivo Excel.

pd.ExcelFile(caminho que está o seu arquivo)

```
xlsx = pd.ExcelFile('/content/drive/My Drive/2.Worksops/Python_Uusarias_Excel/insurance.xlsx')
```

O atributo *sheet_names* retorna os nomes das abas do arquivo Excel.

nome_objeto.sheet_names

```
xlsx.sheet_names
```

```
['insurance', 'insurance2']
```

pyladies São Paulo

Importando múltiplas abas

O método *parse* retorna os dados da aba que desejamos.

nome_objeto.parse(nome_da_aba)

```
df1 = xlsx.parse('insurance')
df1.head(3)
```

	id	age	sex	bmi	children	smoker	region	charges
0	1	19	female	27.90	0	yes	southwest	168849.24
1	2	18	male	33.77	1	no	southeast	172555.23
2	3	28	male	33.00	3	no	southeast	44494.62

```
df2 = xlsx.parse('insurance2')
df2.head(3)
```

	id	ID	country
0	1	1	Brazil
1	2	2	Brazil
2	3	3	Brazil

pyladies São Paulo

Resumo dos dados

Nesta etapa, vamos juntas aprender como visualizar algumas informações importantes do *dataframe*.

pyladies São Paulo

Resumo dos dados do arquivo

Os métodos ***head*** e ***tail*** fazem parte da biblioteca **Pandas**, eles nos auxiliam a ver uma parte do *dataframe*.

O método ***head*** nos retorna por padrão as 5 primeiras linhas, mas podemos alterar este valor também. Já o método ***tail*** por padrão nos retorna as 5 últimas linhas.

```
dados.head()
```

	Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
0	17	AGENTE DE ORGANIZACAO ESCOLAR	EDUCACAO	1549390	774690.0	0.0	0.0	69030.0	0.0	2106490.0
1	30	PROFESSOR EDUCACAO BASICA I	EDUCACAO	3517150	1758570.0	0.0	0.0	512920.0	0.0	4106750.0
2	49	PROFESSOR EDUCACAO BASICA II	EDUCACAO	2959130	0.0	0.0	0.0	0.0	0.0	1316320.0
3	63	PROFESSOR EDUCACAO BASICA II	EDUCACAO	3902780	0.0	0.0	0.0	0.0	0.0	2897890.0
4	65	OFICIAL ADMINISTRATIVO	SAUDE	2434830	0.0	0.0	0.0	392870.0	0.0	2386210.0

nome_dataframe.head()

```
dados.tail()
```

	Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
299801	1048532	PROFESSOR EDUCACAO BASICA I	EDUCACAO	5190870	0.0	0.0	0.0	570990.0	0.0	4458820.0
299802	1048538	AGENTE DE ORGANIZACAO ESCOLAR	EDUCACAO	1662140	0.0	0.0	0.0	173870.0	0.0	1585700.0
299803	1048541	PROFESSOR EDUCACAO BASICA II	EDUCACAO	6188690	0.0	0.0	0.0	631290.0	0.0	3227480.0
299804	1048551	MEDICO II	SAUDE	6575080	0.0	0.0	0.0	723250.0	0.0	5473300.0
299805	1048573	PROFESSOR EDUCACAO BASICA II	EDUCACAO	4336260	0.0	0.0	0.0	0.0	0.0	3448160.0

nome_dataframe.tail()

pyladies São Paulo

Resumo dos dados do arquivo

O método *info* retorna algumas informações úteis, como os tipos de dados por coluna e quantidade de linhas e colunas do *dataframe.*

```
dados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299806 entries, 0 to 299805
Data columns (total 10 columns):
Id                  299806 non-null int64
job                 299806 non-null object
sector              299806 non-null object
Month_salary        299806 non-null int64
13_salary           299806 non-null float64
eventual_salary     299806 non-null float64
indemnity           299806 non-null float64
extra_salary        299806 non-null float64
discount_salary     299806 non-null float64
total_salary        299806 non-null float64
dtypes: float64(6), int64(2), object(2)
memory usage: 22.9+ MB
```

nome_dataframe.info()

pyladies São Paulo

Resumo dos dados do arquivo

No método *unique* os itens únicos são retornados em ordem de aparecimento.

```
dados.sector.unique()
```

```
array(['EDUCACAO', 'SAUDE', 'UNIVESP', 'JUSTICA'], dtype=object)
```

nome_dataframe.nome_coluna.unique()

pyladies São Paulo

Visualizando os dados do arquivo

O método ***shape*** retorna as dimensões do *dataframe*.

```
dados.shape
```

```
(299806, 10)
```

nome_dataframe.shape

O método ***columns*** retorna uma lista com os nomes das colunas do *dataframe*.

```
dados.columns
```

```
Index(['Id', 'job', 'sector', 'Month_salary', '13_salary', 'eventual_salary',
       'indemnity', 'extra_salary', 'discount_salary', 'total_salary'],
      dtype='object')
```

nome_dataframe.columns

pyladies São Paulo

Agora é com você! – Desafio 2

- ✔ Queremos visualizar apenas os 15 primeiros registros do *dataframe*.
- ✔ Queremos visualizar apenas os 20 últimos registros do *dataframe*.

pyladies São Paulo

Resposta - Desafio 2

- ✔ Queremos visualizar apenas os 15 primeiros registros do *dataframe*.

```
dados.head(15)
```

- ✔ Queremos visualizar apenas os 20 últimos registros do *dataframe*.

```
dados.tail(20)
```

pyladies São Paulo

Filtrando os dados

Nesta etapa, vamos juntas aprender como filtrar os dados do *dataframe*.

pyladies São Paulo

Filtrando uma única coluna

Podemos escolher uma única coluna para visualizar.

```
dados['sector']
```

```
0          EDUCACAO
1          EDUCACAO
2          EDUCACAO
3          EDUCACAO
4             SAUDE
              ...
299801     EDUCACAO
299802     EDUCACAO
299803     EDUCACAO
299804        SAUDE
299805     EDUCACAO
Name: sector, Length: 299806, dtype: object
```

nome_dataframe[‘coluna’]

Lembrando que a coluna do *dataframe* é uma series.

pyladies São Paulo

Operadores de Comparação

Em alguns momentos precisamos usar operadores de comparação para fazer filtros.

Abaixo tabela dos operadores de comparação em Python:

Operadores	
Maior que	>
Menor que	<
Maior ou igual	=>
Menor ou igual	<=
Igualdade	==
Diferente	!=

pyladies São Paulo

Filtrando com operadores de comparação

Usando os operadores de comparação para filtrar uma informação específica.

nome_dataframe[‘coluna’] == ‘condição’

```
# Filtrando uma informação específica
dados['sector'] == 'EDUCACAO'
```

```
0          True
1          True
2          True
3          True
4         False
          ...
299801     True
299802     True
299803     True
299804    False
299805     True
Name: sector, Length: 299806, dtype: bool
```

Aqui você coloca o operador lógico que atende o filtro que você precisa.

Aqui você insere a condição para o filtro que você quer. Se a condição for um texto, não se esqueça das aspas!

pyladies São Paulo

Comparação de filtros

Aqui vamos comparar alguns modos de fazer filtro com o Pandas.

```
dados['sector']
```

```
0          EDUCACAO
1          EDUCACAO
2          EDUCACAO
3          EDUCACAO
4             SAUDE
             ...
299801     EDUCACAO
299802     EDUCACAO
299803     EDUCACAO
299804        SAUDE
299805     EDUCACAO
Name: sector, Length: 299806, dtype: object
```

*** Sem operadores de comparação**

```
dados['sector'] == 'EDUCACAO'
```

```
0           True
1           True
2           True
3           True
4          False
           ...
299801      True
299802      True
299803      True
299804     False
299805      True
Name: sector, Length: 299806, dtype: bool
```

*** Com operadores de comparação**

pyladies São Paulo

Filtrando com operadores de comparação

```
nome_dataframe[nome_dataframe['coluna'] == condição]
```

```
dados[dados['job'] == 'PROFESSOR EDUCACAO BASICA I']
```

	Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
1	30	PROFESSOR EDUCACAO BASICA I	EDUCACAO	3517150	1758570.0	0.0	0.0	512920.0	0.0	4106750.0
5	76	PROFESSOR EDUCACAO BASICA I	EDUCACAO	965480	0.0	0.0	0.0	175070.0	0.0	1063320.0
31	307	PROFESSOR EDUCACAO BASICA I	EDUCACAO	2606300	1303140.0	0.0	0.0	132420.0	0.0	2873690.0
36	331	PROFESSOR EDUCACAO BASICA I	EDUCACAO	1421810	0.0	42980.0	0.0	188290.0	0.0	1154480.0
46	383	PROFESSOR EDUCACAO BASICA I	EDUCACAO	2298790	-14970.0	-230950.0	0.0	150870.0	0.0	2008490.0
...	...	...	...	...	...	...	...	...	...	...
299778	1048385	PROFESSOR EDUCACAO BASICA I	EDUCACAO	2718940	0.0	0.0	0.0	82860.0	0.0	1307160.0
299786	1048427	PROFESSOR EDUCACAO BASICA I	EDUCACAO	2748780	0.0	0.0	0.0	139070.0	0.0	2585230.0
299790	1048447	PROFESSOR EDUCACAO BASICA I	EDUCACAO	2298780	0.0	0.0	0.0	108070.0	0.0	2177770.0
299795	1048489	PROFESSOR EDUCACAO BASICA I	EDUCACAO	2336470	0.0	0.0	0.0	163810.0	0.0	2230870.0
299801	1048532	PROFESSOR EDUCACAO BASICA I	EDUCACAO	5190870	0.0	0.0	0.0	570990.0	0.0	4458820.0

50373 rows × 10 columns

Aqui você insere a condição para o filtro que você quer. Se a condição for um texto, não se esqueça das aspas!

Aqui você coloca o operador lógico que atende o filtro que você precisa.

pyladies São Paulo

Filtrando com operadores de lógicos

Tabela dos operadores lógicos em Python:

- & (and/ E)
- | (or/ ou)
- not (não)

pyladies São Paulo

Filtrando com operadores lógicos

Filtrando com duas condições, usando operadores lógicos.

nome_dataframe[(nome_dataframe['coluna'] == 'condição') operador lógico
[(nome_dataframe['coluna'] == 'condição')]

```
dados[(dados['job'] == 'PROFESSOR EDUCACAO BASICA I') & (dados['discount_salary'] > 0)]
```

Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary

Aqui você insere a condição para o filtro que você quer. Se a condição for um texto, não se esqueça das aspas!

Aqui você insere o operador lógico que atenda a sua condição do filtro.

Aqui você coloca o operador lógico que atende o filtro que você precisa.

pyladies São Paulo

Filtrando com operadores lógicos

Outro exemplo de filtro:

```
dados[(dados.sector == 'EDUCACAO') |
      (dados.sector == 'SAUDE') |
      (dados.sector == 'JUSTICA')].head()
```

Uma outra maneira de realizar o filtro acima é usando o método *isin*, deixando ele mais limpo:

```
dados[dados.sector.isin(['EDUCACAO', 'SAUDE', 'JUSTICA'])].head()
```

pyladies São Paulo

Filtrando pelos maiores valores

Para filtrar pelas maiores categorias de uma determinada coluna, usamos o método *value_counts.*

```
nome_dataframe.nome_da_coluna.value_counts()
```

```
dados.sector.value_counts().head()
```

```
EDUCACAO    251957
SAUDE        47348
UNIVESP        301
JUSTICA        200
Name: sector, dtype: int64
```

pyladies São Paulo

Agora é com você! – Desafio 3

Criar um filtro usando operadores de comparação.

- ✔ Filtrar a coluna ‘sector’ para área de saúde E a coluna ‘job’ por oficial administrativo.

pyladies São Paulo

Resposta - Desafio 3

Criar um filtro usando operadores de comparação.

- ✔ Filtrar a coluna ‘sector’ para área de saúde E a coluna ‘job’ por oficial administrativo.

```
dados[(dados['sector'] == 'SAUDE') & (dados['job'] == 'OFICIAL ADMINISTRATIVO')]
```

	Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
4	65	OFICIAL ADMINISTRATIVO	SAUDE	2434830	0.0	0.0	0.0	392870.0	0.0	2386210.0
69	499	OFICIAL ADMINISTRATIVO	SAUDE	1611760	0.0	0.0	0.0	0.0	0.0	1367530.0
91	620	OFICIAL ADMINISTRATIVO	SAUDE	2208780	262290.0	0.0	0.0	234950.0	0.0	1974050.0
99	642	OFICIAL ADMINISTRATIVO	SAUDE	2563920	0.0	0.0	0.0	347510.0	0.0	2369580.0
297	1632	OFICIAL ADMINISTRATIVO	SAUDE	2227410	0.0	0.0	0.0	0.0	0.0	1571890.0
...	...	...	...	...	...	...	...	...	...	...
299440	1047167	OFICIAL ADMINISTRATIVO	SAUDE	1694000	0.0	0.0	0.0	222560.0	0.0	1438020.0
299495	1047336	OFICIAL ADMINISTRATIVO	SAUDE	2653720	0.0	0.0	0.0	173440.0	0.0	1914590.0
299567	1047594	OFICIAL ADMINISTRATIVO	SAUDE	2487070	0.0	0.0	0.0	80840.0	0.0	1624280.0
299618	1047736	OFICIAL ADMINISTRATIVO	SAUDE	2730100	0.0	0.0	0.0	207730.0	0.0	2496700.0
299767	1048294	OFICIAL ADMINISTRATIVO	SAUDE	2521560	0.0	590000.0	0.0	56170.0	0.0	2813900.0

3991 rows × 10 columns

pyladies São Paulo

Criar e excluir colunas

Nesta etapa, vamos juntas aprender como criar e excluir colunas no *dataframe*.

pyladies São Paulo

Operadores matemáticos

Tabela dos operadores matemáticos em Python:

- + (adição)
- - (subtração)
- * (multiplicação)
- / (divisão)
- // (divisão inteira)
- ** (exponencial)
- % (modulo)

pyladies São Paulo

Criando uma nova coluna

Criando uma nova coluna no nosso *dataframe*.

nome_dataframe[‘nome_nova_coluna] = nome_dataframe[‘nome_coluna’] operador matemático com o valor.

```
dados['total_salario'] = dados['Month_salary'] + dados['extra_salary']
dados.head(3)
```

	Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary	total_salario
0	1	OFICIAL ADMINISTRATIVO	DETRAN	2315.81	0.0	0.0	0.0	73.85	0.0	1929.34	2389.66
1	2	SD 2C PM	PM	3034.05	0.0	0.0	0.0	651.82	0.0	2265.96	3685.87
2	3	1TEN PM	PM	8990.98	0.0	0.0	0.0	626.75	0.0	6933.04	9617.73

pyladies São Paulo

Excluindo uma coluna

Excluindo uma coluna no nosso *dataframe*.

nome_dataframe.drop(columns='nome_coluna', axis = 1)

```
dados.drop(columns='total_salario', axis = 1)
```

axis = 1 (coluna)
axis = 0 (indice)

pyladies São Paulo

Excluindo uma coluna - Inplace

Excluindo uma coluna no nosso *dataframe* com o argumento inplace para excluir a coluna.

nome_dataframe.drop(columns='nome_coluna', axis = 1, inplace = True)

```
dados.drop(columns='total_salario', axis = 1, inplace= True)
```

inplace = True
inplace = False

pyladies São Paulo

Agora é com você! – Desafio 4

- Excluir a coluna ‘age’ do *dataframe*, mas sem usar o atributo ***inplace***.

pyladies São Paulo

Resposta - Desafio 4

- ✔ Excluir a coluna '**indemnity**' do *dataframe*, mas sem usar o atributo *inplace*.

```
dados.drop(columns='indemnity', axis = 1)
dados.head()
```

	Id	job	sector	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
0	1	OFICIAL ADMINISTRATIVO	DETRAN	2315.81	0.0	0.0	0.0	73.85	0.0	1929.34
1	2	SD 2C PM	PM	3034.05	0.0	0.0	0.0	651.82	0.0	2265.96
2	3	1TEN PM	PM	8990.98	0.0	0.0	0.0	626.75	0.0	6933.04
3	4	MAJ PM	SPPREV	13591.02	0.0	0.0	0.0	0.00	0.0	10568.36
4	5	AG.TEC. DE ASSIT. A SAUDE	HCFMUSP	4203.67	0.0	0.0	0.0	0.00	0.0	3561.88

pyladies São Paulo

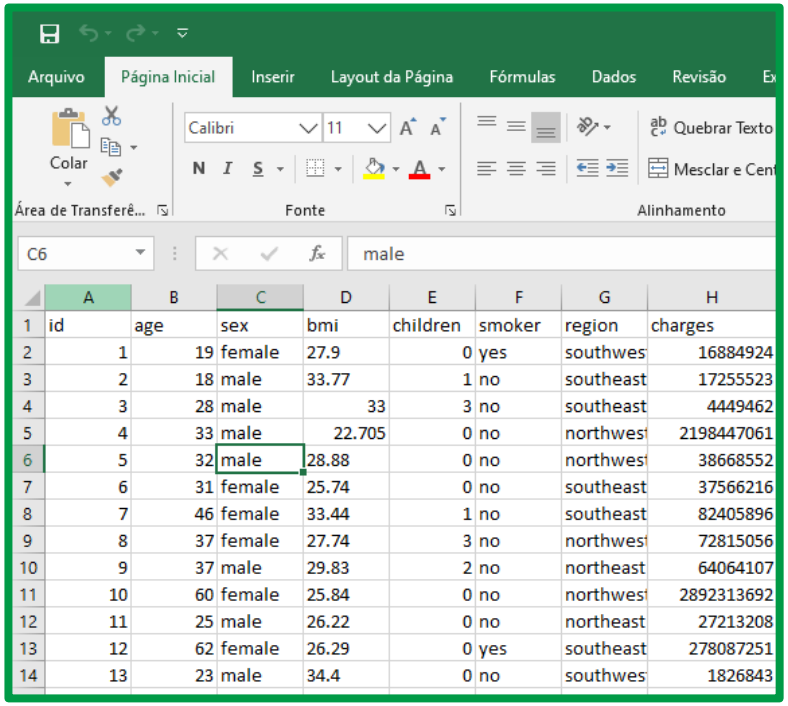
Tabelas Dinâmicas

Nesta etapa, vamos juntas aprender como criar tabelas dinâmicas.

pyladies São Paulo

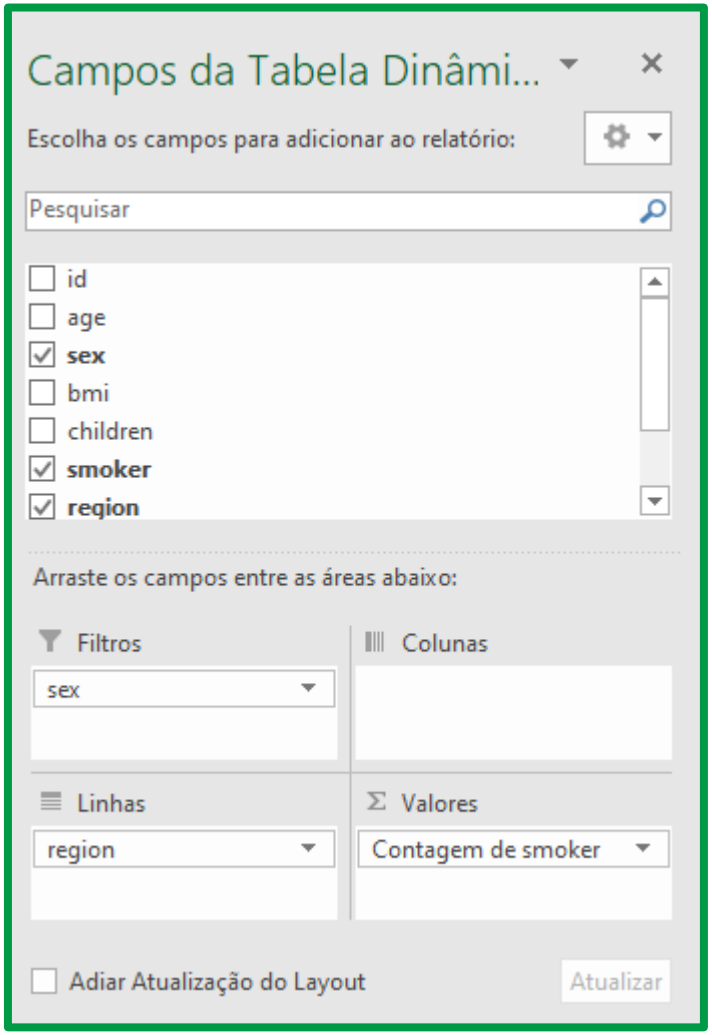
Tabela Dinâmica (Pivot Table)

Raw Data



	A	B	C	D	E	F	G	H
1	id	age	sex	bmi	children	smoker	region	charges
2	1	19	female	27.9	0	yes	southwes	16884924
3	2	18	male	33.77	1	no	southeast	17255523
4	3	28	male	33	3	no	southeast	4449462
5	4	33	male	22.705	0	no	northwest	2198447061
6	5	32	male	28.88	0	no	northwest	38668552
7	6	31	female	25.74	0	no	southeast	37566216
8	7	46	female	33.44	1	no	southeast	82405896
9	8	37	female	27.74	3	no	northwest	72815056
10	9	37	male	29.83	2	no	northeast	64064107
11	10	60	female	25.84	0	no	northwest	2892313692
12	11	25	male	26.22	0	no	northeast	27213208
13	12	62	female	26.29	0	yes	southeast	278087251
14	13	23	male	34.4	0	no	southwes	1826843

Editor da Tabela Dinâmica



	A	B
1	sex	(Tudo)
2		
3	Rótulos de Linha	Contagem de smoker
4	northeast	324
5	northwest	325
6	southeast	364
7	southwest	325
8	Total Geral	1338

Tabela Dinâmica

pyladies São Paulo

Tabela Dinâmica - Sum

Usamos o método *sum* para somar as colunas numéricas.

nome_dataframe.sum(numeric_only=True)

```
dados.sum(numeric_only=True)
```

```
Id                 5.888714e+11
Month_salary       4.719238e+09
13_salary          2.352580e+08
eventual_salary    1.483163e+08
indemnity          1.917648e+07
extra_salary       1.260761e+08
discount_salary   -1.032641e+08
total_salary       3.682726e+09
dtype: float64
```

pyladies São Paulo

Tabela Dinâmica - Groupby

Usamos o método *groupby* para agrupar os valores por alguma coluna, ex.: agrupando os valores por sector.

nome_dataframe.groupby(by='nome_da_coluna', as_indice=False).sum()

```
dados.groupby(by='sector', as_index=False).sum()
```

	sector	Id	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
0	ADM GERAL	17433579560	1.672113e+08	3815195.59	3331121.05	0.00	2014.23	-4806746.26	86480223.47
1	AGEM	9383056	8.957459e+04	6626.41	3085.97	0.00	95.16	0.00	78311.37
2	AGEMCAMP	7808591	9.269032e+04	3767.48	0.00	0.00	0.00	0.00	71576.32
3	AGEMVALE	1092933	3.074369e+04	2166.13	2114.58	0.00	0.00	0.00	26936.78
4	ARSESP	78147779	1.120013e+06	102335.30	-5057.64	0.00	0.00	-1317.12	826590.84

pyladies São Paulo

Tabela Dinâmica - Groupby Múltiplas Colunas

Usamos o método *groupby* para agrupar os valores com múltiplas colun as, ex.: agrupando valores por sector e job.

nome_dataframe.groupby(by=[‘nome_da_coluna’], [‘nome_da_coluna’], as_indice=False).sum()

```
dados.groupby(by=['sector', 'job'], as_index=False).sum()
```

	sector	job	Id	Month_salary	13_salary	eventual_salary	indemnity	extra_salary	discount_salary	total_salary
0	ADM GERAL	BENEFICIARIOS-PENSAO PARLAMENTAR	100932914	1855257.65	0.00	31336.27	0.0	0.00	-11073.60	1442775.87
1	ADM GERAL	COMPL.APOSENTADORIA - FERROBAN (FEPASA)	4960694	90411.94	0.00	0.00	0.0	0.00	0.00	46992.63
2	ADM GERAL	COMPL.PENSAO - FERROBAN (FEPASA)	809956	8913.13	0.00	0.00	0.0	0.00	0.00	5434.54
3	ADM GERAL	COMPLEM APOSENT - BANCO NOSSA CAIXA	47622962	1001869.48	27567.69	0.00	0.0	0.00	-74484.00	533604.33
4	ADM GERAL	COMPLEM APOSENT-BANCO NOSSA CAIXA	17823070	285062.93	24532.19	-132.97	0.0	0.00	-12301.83	221601.78

pyladies São Paulo

O que é uma lista? Usando lista com Group By

Usando uma lista com nomes de colunas que queremos agrupar.

Mas o que é uma lista? Uma lista (list) em Python é uma sequência ou coleção ordenada de valores.

nome_lista = [‘nome’,’nome’,…, ’nome’]

```
lista = ['sector', 'job']
```

nome_dataframe.groupby(by=nome_lista, as_indice=False).sum()

```
dados.groupby(by=lista, as_index=False).sum().head(3)
```

pyladies São Paulo

Tabela Dinâmica - Groupby + Agg

Usamos o *groupby* com a função *agg*, pois usando as duas consegui mos definir qual coluna vamos agrupar os dados e com coluna vamos usar para ter os valores.

```
nome_dataframe.groupby('coluna')['coluna'].agg(['método', 'método'])
```

```
dados.groupby('sector')['Month_salary'].agg(['mean', 'count'])
```

sector	mean	count
ADM GERAL	5043.169843	33156
AGEM	4976.366111	18
AGEMCAMP	6620.737143	14
AGEMVALE	7685.922500	4
ARSESP	8116.037899	138

pyladies São Paulo

Tabela Dinâmica - Pivot Table

No Pandas temos uma função chamada *Pivot Table* que é exatamente o que fazemos no Excel.

nome_dataframe.pivot_table(index=”coluna_index”, columns=”coluna”, values=”coluna_dos_valores”, aggfunc = “operacao matematica)

```
pvt.pivot_table(index = 'job', columns='sector', values='total_salary', aggfunc='count')
```

job	sector	EDUCACAO	JUSTICA	SAUDE
AG.APOIO PESQ.CIENT.TECNOL.		NaN	NaN	61.0
AG.TEC. DE ASSIT. A SAUDE		46.0	NaN	2817.0
AGENTE DE ORGANIZACAO ESCOLAR		33516.0	NaN	NaN
AGENTE DE SANEAMENTO		NaN	NaN	122.0
AGENTE DE SAUDE		NaN	NaN	331.0

pyladies São Paulo

Tabela Dinâmica - Cross Table

A função *crosstab* realiza uma transposição dos dados.

pd.crosstab(nome_dataframe.nome_coluna, nome_dataframe.nome_coluna)

```
pd.crosstab(pvt.sector,pvt.job)
```

job sector	AG.APOIO PESQ.CIENT.TECNOL.	AG.TEC. DE ASSIT. A SAUDE	AGENTE DE ORGANIZACAO ESCOLAR	AGENTE DE SANEAMENTO	AGENTE DE SAUDE	AGENTE DE SERVICOS ESCOLARES	AGENTE TECNICO DE SAUDE	ANALISTA ADMINISTRATIVO	ANALISTA DE DADOS II	ANALISTA DE O M V	ANALISTA DE SISTEMAS IV	ANALISTA SOCIO CULTURAL	ANALISTA SUPERVISOR
EDUCACAO	0	46	33516	0	0	5464	0	364	0	0	0	49	0
JUSTICA	0	0	0	0	0	0	0	0	0	0	0	1	0
SAUDE	61	2817	0	122	331	0	240	30	1	1	2	23	1

3 rows × 128 columns

pyladies São Paulo

VlookUp

Nesta etapa, vamos juntas aprender como realizar VlookUp.

pyladies São Paulo

VlookUp - Merge

O *vlookup* pode ser feito no pandas com o método *merge* e com o argumento *on*, o argumento *on* especifica a coluna que vamos usar para dar o *match* das colunas.

nome_dataframe.merge(nome_segundo_dataframe, on = “nome_coluna”)

```
df1.merge(df2, on='id', how='left').head(3)
```

	id	age	sex	bmi	children	smoker	region	charges	ID	country
0	1	19	female	27.90	0	yes	southwest	168849.24	1	Brazil
1	2	18	male	33.77	1	no	southeast	172555.23	2	Brazil
2	3	28	male	33.00	3	no	southeast	44494.62	3	Brazil

pyladies São Paulo

Formatação

Nesta etapa, vamos juntas aprender como formatar os dados.

pyladies São Paulo

Formatação

Nesta etapa, vamos aprender a formatar as colunas e os dados do nosso *dataframe*.

O pandas possui um sistema de opções que permite personalizar alguns aspectos de seu comportamento, sendo as opções relacionadas à exibição dos dados que o usuário provavelmente ajustará.

pd.set_option(display.o_qual_tipo_quer_mudar, o_que_quer_mudar)

```
pd.set_option('display.float_format', '{:.1f}'.format)
```

pyladies São Paulo

Formatação - set option

Nesta etapa, vamos aprender a formatar as colunas e os dados do nosso *dataframe*.

O pandas possui um sistema de opções que permite personalizar alguns aspectos de seu comportamento, sendo as opções relacionadas à exibição dos dados que o usuário provavelmente ajustará.

```
pd.set_option('display.float_format', '{:.1f}'.format)
```

E para voltar a configuracao anterior usamos o *reset_option.*

```
pd.reset_option('display.float_format')
```

pyladies São Paulo

Formatação - str

Para formatar texto no Python usamos o atributo *str*.

Se usarmos o atributo *str* + o método *title* conseguimos transformar o texto da variável para o padrão de começar com letra maiúscula.

nome_dataframe.str.title()

```
df1['region'] = df1['region'].str.title()
```

pyladies São Paulo

Formatação - str

Usando o atributo *str* + o método *upper* conseguimos transformar o texto da variável para o padrão de deixar as letras maiúsculas.

nome_dataframe.str.upper()

```
df1['region'] = df1['region'].str.upper()
```

pyladies São Paulo

Formatação - str

Usando o atributo *str* + o método *lower* conseguimos transformar o texto da variável para o padrão de deixar as letras minúsculas.

nome_dataframe.str.lower()

```
df1['region'] = df1['region'].str.lower()
```

pyladies São Paulo

Formatação - str

Usando o atributo *str* + o método *strip* remove os espaços em branco que tiver no texto.

nome_dataframe.str.strip()

```
df1['region'] = df1['region'].str.strip()
```

pyladies São Paulo

Formatação - rename

Usando o método *rename* conseguimos mudar o nome das variáveis.

nome_dataframe.rename(columns = {‘nome_coluna’: ‘nome_novo’})

```
df1.rename(columns = {'region': 'região', 'age':'idade'}).head(3)
```

pyladies São Paulo

Formatação - cut

Use a função *cut* quando precisar segmentar e classificar valores de dados em compartimentos.

Essa função também é útil para passar de uma variável contínua para uma variável categórica. Por exemplo, pode converter idades em grupos de faixas etárias.

pd.cut(nome_coluna_dados, bins=[qtd de cortes], labels=[‘nome_dos_cortes’]

```
df1['categoria_idade'] = pd.cut(df1.age, bins=[0, 18, 50, 99], labels=['crianças', 'jovens', 'adultos'])
```

pyladies São Paulo

Formatação Condicional

Nesta etapa, vamos juntas aprender como usar a formatação condicional dos dados.

pyladies São Paulo

Formatação Condicional

Aqui vamos destacar a célula com o valor minímo e máximo da variável 'bmi'.

```
pd.set_option('display.float_format', '{:.1f}'.format)
```

```
(df1.style.format(format_dict)
 .hide_index()
 .highlight_min('bmi', color='red')
 .highlight_max('bmi', color='lightgreen')
)
```

id	age	sex	bmi	children	smoker	region	charges	categoria_idade
1	19	female	27.90	0	yes	southwest	168849	jovens
2	18	male	33.77	1	no	southeast	172555	crianças
3	28	male	33.00	3	no	southeast	44494.6	jovens
4	33	male	22705.00	0	no	northwest	2.19845e+07	jovens
5	32	male	28.88	0	no	northwest	386686	jovens
6	31	female	25.74	0	no	southeast	375662	jovens
7	46	female	33.44	1	no	southeast	824059	jovens
8	37	female	27.74	3	no	northwest	728151	jovens
9	37	male	29.83	2	no	northeast	640641	jovens
10	60	female	25.84	0	no	northwest	2.89231e+07	adultos
11	25	male	26.22	0	no	northeast	272132	jovens
12	62	female	26.29	0	yes	southeast	2.78087e+06	adultos
13	23	male	34.40	0	no	southwest	18268.4	jovens
14	56	female	39.82	0	no	southeast	1.10907e+06	adultos
15	27	male	42.13	0	yes	southeast	3.96118e+06	jovens

pyladies São Paulo

Formatação Condicional

Aqui vamos destacar a variável 'age' em degrade.

```
(df1.style.format(format_dict)
 .hide_index()
 .background_gradient(subset='age', cmap='Blues')
)
```

id	age	sex	bmi	children	smoker	region	charges	categoria_idade
1	19.00	female	27.9	0	yes	southwest	168849	jovens
2	18.00	male	33.77	1	no	southeast	172555	crianças
3	28.00	male	33	3	no	southeast	44494.6	jovens
4	33.00	male	22705	0	no	northwest	2.19845e+07	jovens
5	32.00	male	28.88	0	no	northwest	386686	jovens
6	31.00	female	25.74	0	no	southeast	375662	jovens
7	46.00	female	33.44	1	no	southeast	824059	jovens
8	37.00	female	27.74	3	no	northwest	728151	jovens
9	37.00	male	29.83	2	no	northeast	640641	jovens
10	60.00	female	25.84	0	no	northwest	2.89231e+07	adultos
11	25.00	male	26.22	0	no	northeast	272132	jovens
12	62.00	female	26.29	0	yes	southeast	2.78087e+06	adultos
13	23.00	male	34.4	0	no	southwest	18268.4	jovens
14	56.00	female	39.82	0	no	southeast	1.10907e+06	adultos
15	27.00	male	42.13	0	yes	southeast	3.96118e+06	jovens

pyladies São Paulo

Gráfico

Nesta etapa, vamos juntas aprender como gerar gráficos.

pyladies São Paulo

Gráfico - Bibliotecas

Para gerar gráficos em Python usamos as bibliotecas **seaborn** e **matplotlib**.

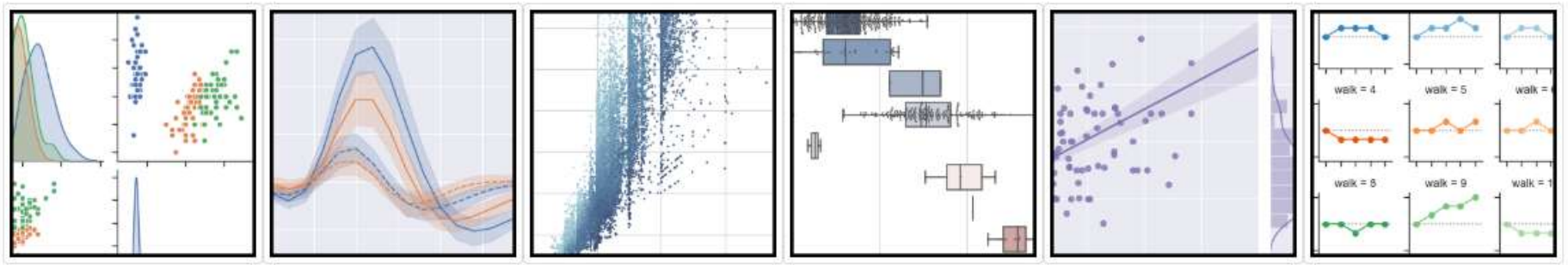
```
import seaborn as sns
import matplotlib.pyplot as plt
```

pyladies São Paulo

Gráfico - Bibliotecas

Seaborn é uma biblioteca de visualização de dados Python baseada no matplotlib.

Ele fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos.



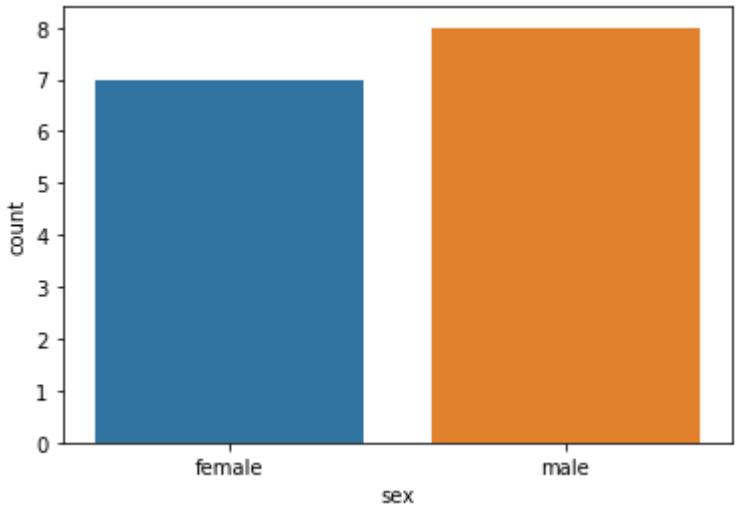
pyladies São Paulo

Gráfico - Countplot

Vamos usar o *countplot* que é um tipo de gráfico, ele conta a quantidade de vezes que aparece uma determinada observação de texto.

sns.tipo_gráfico(x='variavel_escolhida', data='nome_dataframe')

plt.show()




```
sns.countplot(x='sex', data=df1)
plt.show()
```

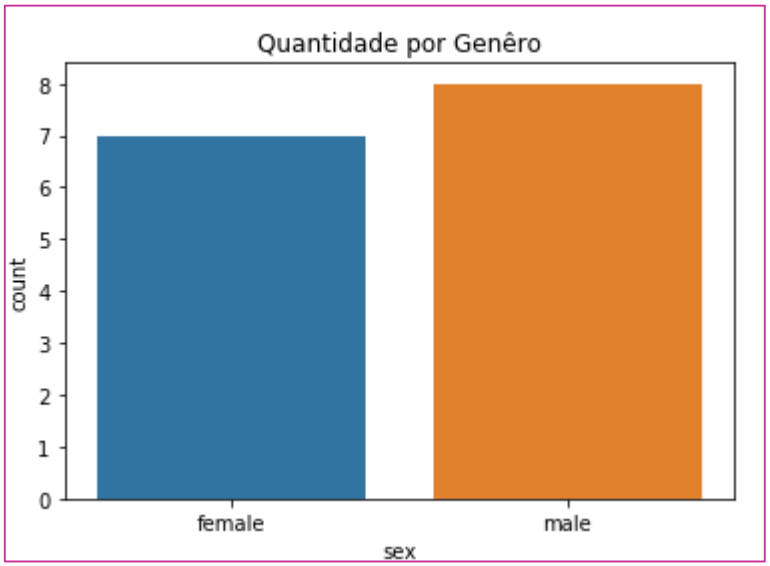
pyladies São Paulo

Gráfico - Formatação

Podemos colocar algumas informações no gráfico, como título.

sns.tipo_gráfico(x='variavel_escolhida', data='nome_dataframe')

plt.show()
plt.title(nome_título')




```
sns.countplot(x='sex', data=df1)
plt.title('Quantidade por Genêro')
plt.show()
```

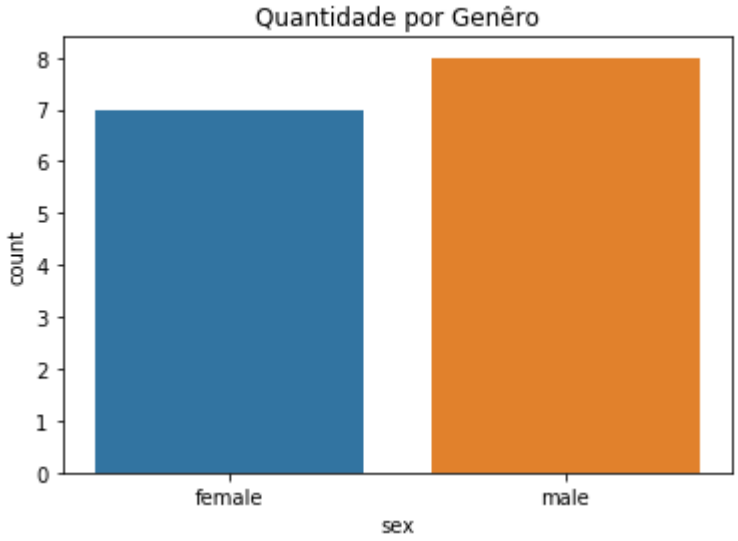
pyladies São Paulo

Gráfico - Formatação

Podemos colocar algumas informações no gráfico, como título.

```
sns.tipo_gráfico(x='variavel_escolhida', data='nome_dataframe')
```

```
plt.show()
plt.title(nome_título')
```




```
sns.countplot(x='sex', data=df1)
plt.title('Quantidade por Genêro')
plt.show()
```

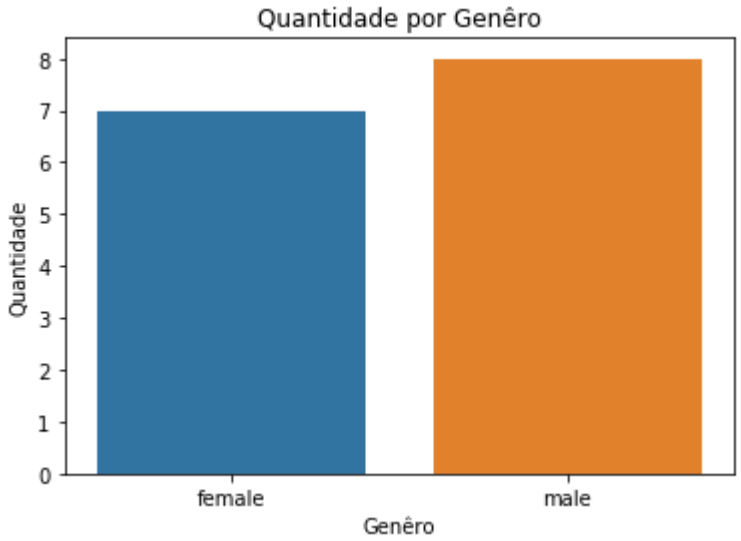
pyladies São Paulo

Gráfico - Formatação

Podemos colocar também nomes nos eixos x e y.

```
sns.tipo_gráfico(x='variavel_escolhida',
data='nome_dataframe')
```

```
plt.title(nome_título')
plt.xlabel('nome_eixo_x')
plt.ylabel('nome_eixo_y')
plt.show()
```




```
sns.countplot(x='sex', data=df1)
plt.title('Quantidade por Genêro')
plt.xlabel('Genêro')
plt.ylabel('Quantidade')
plt.show()
```

pyladies São Paulo

O que vimos hoje

1. Tipo de Dados
2. Estrutura de Dados
3. Importar dados para o Python
4. Importar múltiplas abas para o Python
5. Resumo/ Visualização dos dados
6. Filtrando os dados
7. Criar e excluir colunas
8. Tabelas Dinâmicas/ Pivot Table
9. ProcV/ Vlookup
10. Formatação dos dados
11. Gráfico

pyladies São Paulo

Bibliografia

Utilizei como base de aprendizado alguns caminhos que irei compartilhar abaixo:

https://github.com/ank0409/Ditching-Excel-for-Python/blob/master/Ditching%20Excel%20for%20Python!.ipynb

https://github.com/PyLadiesSP/data-science/tree/master/workshops/workshop_introdu%C3%A7%C3%A3o_estatistica_pandas

https://www.kaggle.com/gustavomodelli/monthly-salary-of-public-worker-in-brazil#monthly_salary_brazil.csv

https://github.com/justmarkham/pandas-videos/blob/master/top_25_pandas_tricks.ipynb

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

https://towardsdatascience.com/pandas-from-basic-to-advanced-for-data-scientists-aee4eed19cfe

https://www.youtube.com/watch?v=cRELNmDpaks

pyladies São Paulo

E por hoje é só pessoal!

PyLadiesSP

@PyLadiesSP

PyLadiesSP

saopaulo@pyladies.com