



INTENSIVÃO DE PYTHON {#}

100% ONLINE & GRATUITO

Apostila Completa Aula 1

Aprenda criar e enviar relatórios, por e-mail,
automaticamente
Impressionador do absoluto zero!



Parte 1

Introdução

O que vamos aprender

Nas primeira aula do Intensivão do Python você vai aprender a criar um código de automação de análise de dados e elaboração de relatórios do **absoluto zero**. Para isso, vamos passar por conceitos como:

Google Colab

Variáveis, listas,
métodos e funções

Tratar dados usando
a biblioteca Pandas

Importação de
bibliotecas

Enviar e-mails
automaticamente

Após todos esses conhecimentos, seremos capazes de transformar uma tabela cheia de informações, nem um pouco fáceis de serem interpretadas ...

... em uma ferramenta automatizada de geração e envio automático de relatórios para uma lista de e-mail pré-definida

A	B	C	D	E	F	G
Código Vend	Data	ID Loja	Produto	Quantid	Valor Unit	Valor Final
1	01/01/2019	Iguatemi Esplanada	Sapato Estampa	1	R\$ 358,00	R\$ 358,00
1	01/01/2019	Iguatemi Esplanada	Camiseta	2	R\$ 180,00	R\$ 360,00
1	01/01/2019	Iguatemi Esplanada	Sapato Xadrez	1	R\$ 368,00	R\$ 368,00
2	02/01/2019	Norte Shopping	Relógio	3	R\$ 200,00	R\$ 600,00
2	02/01/2019	Norte Shopping	Chinelo Liso	1	R\$ 71,00	R\$ 71,00
3	02/01/2019	Rio Mar Shopping Fortaleza	Cinto Linho	1	R\$ 248,00	R\$ 248,00
5	02/01/2019	Shopping Barra	Calça	1	R\$ 170,00	R\$ 170,00
6	02/01/2019	Shopping Ibirapuera	Polo Listrado	4	R\$ 149,00	R\$ 596,00
7	02/01/2019	Norte Shopping	Camisa Gola V Listrado	1	R\$ 116,00	R\$ 116,00
7	02/01/2019	Norte Shopping	Camisa Liso	1	R\$ 105,00	R\$ 105,00
8	02/01/2019	Iguatemi Campinas	Terno Estampa	5	R\$ 706,00	R\$ 3.530,00
9	02/01/2019	Shopping Center Leste Aricanduva	Tênis Linho	1	R\$ 294,00	R\$ 294,00
9	02/01/2019	Shopping Center Leste Aricanduva	Casaco Xadrez	2	R\$ 259,00	R\$ 518,00
10	02/01/2019	Passei das Águas Shopping	Camiseta Xadrez	2	R\$ 200,00	R\$ 400,00
10	02/01/2019	Passei das Águas Shopping	Cueca Liso	1	R\$ 69,00	R\$ 69,00
11	02/01/2019	Shopping Recife	Camiseta	1	R\$ 180,00	R\$ 180,00
11	02/01/2019	Shopping Recife	Calça Listrado	2	R\$ 181,00	R\$ 362,00
11	02/01/2019	Shopping Recife	Camisa Estampa	1	R\$ 113,00	R\$ 113,00
12	02/01/2019	Shopping Midway Mall	Short Xadrez	4	R\$ 100,00	R\$ 400,00
12	02/01/2019	Shopping Midway Mall	Sunga Liso	2	R\$ 114,00	R\$ 228,00
14	02/01/2019	Bourbon Shopping SP	Casaco Liso	2	R\$ 255,00	R\$ 510,00
17	02/01/2019	Iguatemi Campinas	Chinelo Liso	1	R\$ 71,00	R\$ 71,00
18	02/01/2019	Rio Mar Shopping Fortaleza	Camisa Estampa	1	R\$ 113,00	R\$ 113,00



pythonimpressionador	Caixa de entrada	Loja: Todas as Lojas - Coe Lira, Valor Final Ticket Médio ID Loja Iguatemi Campin...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Salvador Shopping - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Salvador Shop...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping Morumbi - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Shopping Mor...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping Iguatemi Fortaleza - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Sho...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Ribeirão Shopping - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Ribeirão Shop...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping SP Market - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Shopping SP...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Palladium Shopping Curitiba - Coe Lira, Quantidade Valor Final Ticket Médio L...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Rio Mar Recife - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Rio Mar Recife B8...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Novo Shopping Ribeirão Preto - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja No...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping Vila Velha - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Shopping Vil...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping Eldorado - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Shopping Eld...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping União de Osasco - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Shop...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Center Shopping Uberlândia - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Cent...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Parque Dom Pedro Shopping - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Par...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Shopping Center Interlagos - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Shop...	28 de jan.
pythonimpressionador	Caixa de entrada	Loja: Bourbon Shopping SP - Coe Lira, Quantidade Valor Final Ticket Médio ID Loja Bourbon S...	28 de jan.

Entendendo a base de dados

As informações que vão alimentar o nosso código, serão dados referentes a vendas de vários produtos de diferentes lojas.

A imagem ao lado, mostra as primeiras linhas da tabela. As informações que temos em cada uma das colunas são:

- ✓ Código venda
- ✓ Data
- ✓ ID Loja
- ✓ Produto
- ✓ Quantidade
- ✓ Valor Unitário
- ✓ Valor Final

Portanto, para um melhor entendimento, na linha 2 temos o seguinte:

Uma venda com código **1** no dia **01/01/2019**, na loja **IGUATEMI ESPLANADA** do **PRODUTO SAPATO ESTAMPA** em uma quantidade igual a **1**, a um preço de **R\$ 358,00** por produto, totalizando uma venda de **R\$ 358,00**.

Como você pode ver, temos muitas informações de vendas nessa tabela, e qualquer interpretação desses dados não é uma tarefa fácil, o que é um problema.

	A	B	C	D	E	F	G
1	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
2	1	01/01/2019	Iguatemi Esplanada	Sapato Estampa	1	R\$ 358,00	R\$ 358,00
3	1	01/01/2019	Iguatemi Esplanada	Camiseta	2	R\$ 180,00	R\$ 360,00
4	1	01/01/2019	Iguatemi Esplanada	Sapato Xadrez	1	R\$ 368,00	R\$ 368,00
5	2	02/01/2019	Norte Shopping	Relógio	3	R\$ 200,00	R\$ 600,00
6	2	02/01/2019	Norte Shopping	Chinelo Liso	1	R\$ 71,00	R\$ 71,00
7	3	02/01/2019	Rio Mar Shopping Fortaleza	Cinto Linho	1	R\$ 248,00	R\$ 248,00
8	5	02/01/2019	Shopping Barra	Calça	1	R\$ 170,00	R\$ 170,00
9	6	02/01/2019	Shopping Ibirapuera	Polo Listrado	4	R\$ 149,00	R\$ 596,00
10	7	02/01/2019	Norte Shopping	Camisa Gola V Listrado	1	R\$ 116,00	R\$ 116,00
11	7	02/01/2019	Norte Shopping	Camisa Liso	1	R\$ 105,00	R\$ 105,00
12	8	02/01/2019	Iguatemi Campinas	Terno Estampa	5	R\$ 706,00	R\$ 3.530,00
13	9	02/01/2019	Shopping Center Leste Aricanduva	Tênis Linho	1	R\$ 294,00	R\$ 294,00
14	9	02/01/2019	Shopping Center Leste Aricanduva	Casaco Xadrez	2	R\$ 259,00	R\$ 518,00
15	10	02/01/2019	Passei das Águas Shopping	Camiseta Xadrez	2	R\$ 200,00	R\$ 400,00
16	10	02/01/2019	Passei das Águas Shopping	Cueca Liso	1	R\$ 69,00	R\$ 69,00
17	11	02/01/2019	Shopping Recife	Camiseta	1	R\$ 180,00	R\$ 180,00
18	11	02/01/2019	Shopping Recife	Calça Listrado	2	R\$ 181,00	R\$ 362,00
19	11	02/01/2019	Shopping Recife	Camisa Estampa	1	R\$ 113,00	R\$ 113,00
20	12	02/01/2019	Shopping Midway Mall	Short Xadrez	4	R\$ 100,00	R\$ 400,00
21	12	02/01/2019	Shopping Midway Mall	Sunga Liso	2	R\$ 114,00	R\$ 228,00
22	14	02/01/2019	Bourbon Shopping SP	Casaco Liso	2	R\$ 255,00	R\$ 510,00
23	17	02/01/2019	Iguatemi Campinas	Chinelo Liso	1	R\$ 71,00	R\$ 71,00
24	18	02/01/2019	Rio Mar Shopping Fortaleza	Camisa Estampa	1	R\$ 113,00	R\$ 113,00

Entendendo a solução final

Como temos muitos produtos e muitas lojas distintas, temos muito trabalho em emitir manualmente um relatório para cada um dos gerentes de loja....Portanto, uma solução para o problema, é construir um *código* que reduza o nosso trabalho operacional, nos tornando mais eficientes.

Mas o que é um código? Vamos dizer que um código é como uma receita de cozinha para computadores. Nesse código, temos o passo a passo que indica ao computador o que fazer, e como fazer.

O principal objetivo desse código é permitir uma melhor interpretação dos dados, em termos de qualidade e velocidade de análise.

O nosso código disparará e-mails de análise de cada produto de cada uma das lojas para seus respectivos gerentes **automaticamente!** O mais impressionante, que não vai demorar nem 1 minuto!!

Então, vamos começar!

Desafio:

Você faz parte da equipe de Analytics de uma grande marca de vestuário com mais de 25 lojas espalhadas em Shoppings de todo o Brasil.

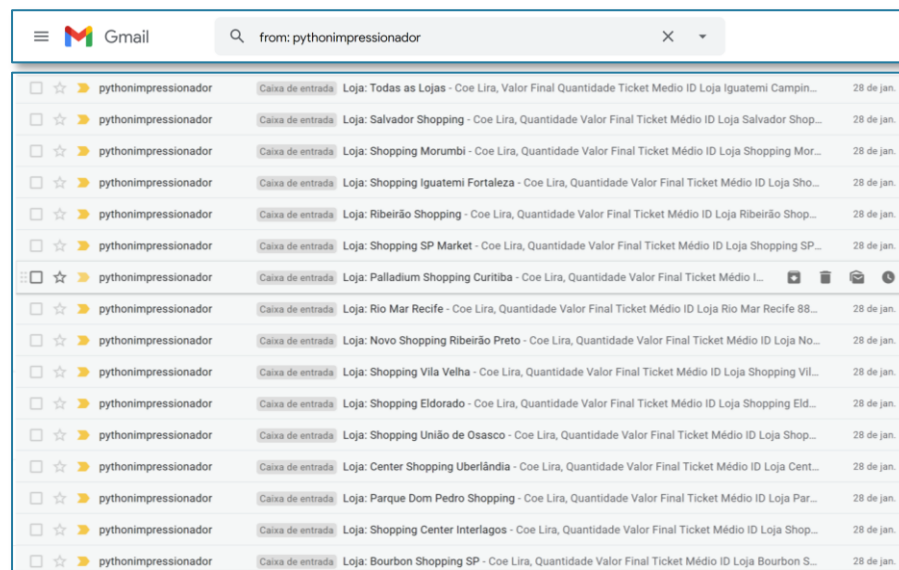
Toda semana você precisa enviar para a diretoria um ranking atualizado com as 25 lojas contendo 3 informações:

- Faturamento de cada Loja
- Quantidade de Produtos Vendidos de cada Loja
- Ticket Médio dos Produto de cada Loja

Além disso, cada loja tem 1 gerente que precisa receber o resumo das informações da loja dele. Por isso, cada gerente deve receber no e-mail:

- Faturamento da sua loja
- Quantidade de Produtos Vendidos da sua loja
- Ticket Médio dos Produto da sua Loja

Esse relatório é sempre enviado como um resumo de todos os dados disponíveis no ano.



Parte 2

O que é o Python

INTENSIVÃO DE {#}
PYTHON{#}
100% ONLINE & GRATUITO

O que é o Python

O Python, é uma linguagem de programação.

Ok.... Mas o que é uma linguagem de programação??

Assim como temos diferentes línguas para falarmos, existem diversas línguas que nos permitem “falar” com os computadores.

Entre as línguas de produção, o Python é uma das mais fáceis de aprender e uma das que mais cresce no mundo em termos de utilização.

Pode ser utilizado em diversas áreas:

- Data Science;
- Automação de processos;
- Desenvolvimento de sites;
- Inteligência artificial;
- Vários outros

Curiosidade: Seu nome apesar de geralmente ser vinculado a cobra, não tem essa origem.... Na verdade, ele é uma homenagem a um grupo de comédia inglês chamado Monty Python 😊



Parte 3

Google Colab

O que é? Como acesso?

Os códigos em Python precisam de uma plataforma para serem escritos.

Essas plataformas na programação são chamadas de IDEs. Existem várias: Visual Studio, PyCharm, Atom, Jupyter Notebook, etc...

Todas podem ser utilizadas pra programação e execução dos códigos, no entanto, a maioria delas necessita de uma fase anterior de instalação que nos tomaria algum tempo...

Por isso, vamos usar a ferramenta Online e Gratuita da Google: Google Colab no intensivão.

Esta ferramenta é muito próxima do Jupyter Notebook e nos permite escrever, testar e executar códigos sem nenhum passo de instalação anterior.

Para acessar basta “dar um Google” por **Google Colab** e clicar no primeiro link!



Preparativos

Se você já possui uma conta @gmail você deverá realizar o Login da sua conta.

Para o intensivão nós vamos realizar 3 etapas:

- 1) Crie uma pasta **Projetos Intensivão de Python** dentro da pasta **Colab Notebooks**;
- 2) Carregar a planilha **Vendas.xlsx** na pasta criada conforme indicado ao lado.

É necessário fazer login no Google

É necessário fazer login com uma Conta do Google para continuar.

CANCELAR

FAZER LOGIN

drive

MyDrive

Colab Notebooks

Projetos Intensivão de Py...

Aula 1

Vendas.xlsx

O que é? Como acesso?

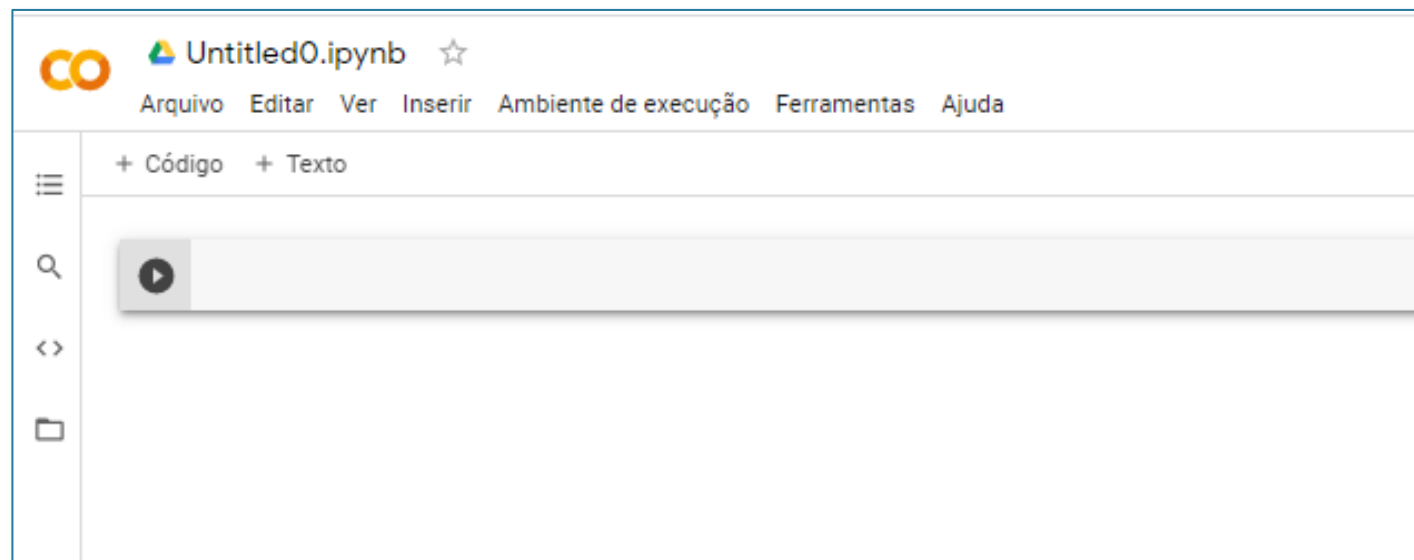
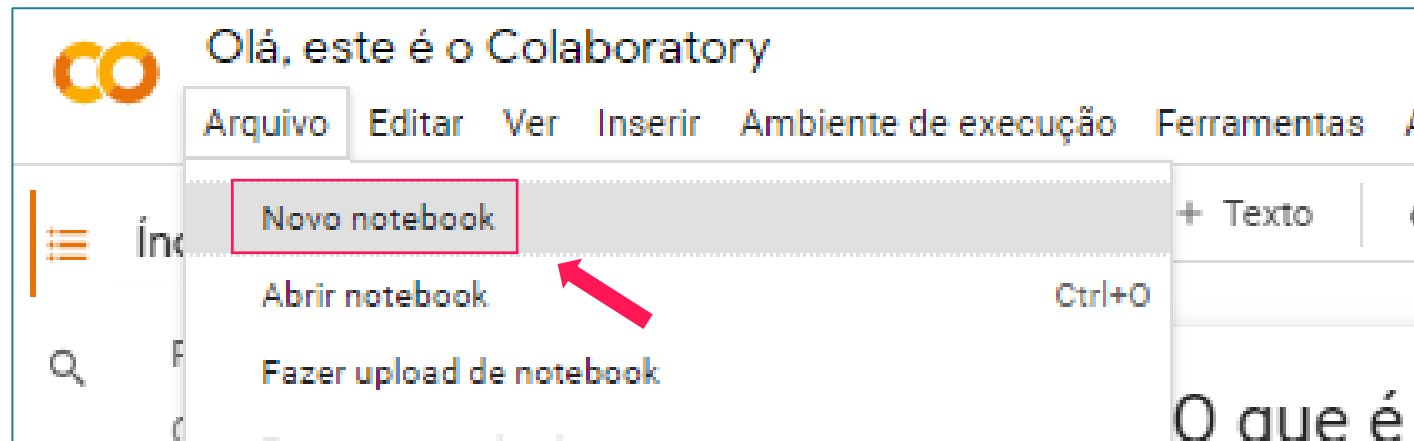
Após os preparativos, vamos acessar o Google Colab para criarmos nosso arquivo que receberá o código em Python.

Esse arquivo se chama Notebook e possui o formato *.ipynb*. Ele só será aberto dentro de plataformas como o Google Colab ou Jupyter Notebook.

Para **criarmos um novo Notebook**, devemos clicar em **Arquivo > Novo Notebook**, assim como apresentado na figura ao lado.

Ao criar o novo Notebook, o Google Colab abrirá a janela ao lado. Aqui, é onde escreveremos nosso programa.

Mas antes de tudo, vamos entender a interface dessa plataforma.



Entendendo a interface

The image shows the Google Colab interface with several components highlighted by red boxes and labeled with blue text boxes:

- Nome do arquivo. Formato ipynb.**: Points to the filename `Untitled0.ipynb` at the top left.
- Arquivo**, **Editar**, **Ver**, **Inserir**, **Ambiente de execução**, **Ferramentas**, **Ajuda**: These are the menu items located below the filename.
- Todas as alterações foram salvas**: A status message on the right side of the menu bar.
- + Código** and **+ Texto**: Buttons in the left sidebar to create new code or text cells. **Cria nova célula de código** points to the **+ Código** button, and **Cria nova célula de texto** points to the **+ Texto** button.
- Célula onde deve ser escrito o código**: Points to the code input area of a cell containing `print('olá')`.
- Executa o código**: Points to the play button icon on the left of the code cell.
- Região de Output**: Points to the output area below the code cell, which displays the text `olá`.

Parte 4

Importando e visualizando os dados

Importando e visualizando os dados

Importando bibliotecas

O Python por si só já possui uma série de funcionalidades que nos permitem ser mais ágeis e eficientes na programação.

No entanto, por se tratar de um código aberto, diversos pacotes de código foram criados para ajudar ainda mais a elaboração dos códigos.

Esses pacotes são chamados de **bibliotecas**.

Aqui vamos importar um dos mais conhecidos, o **PANDAS**([link](#)).

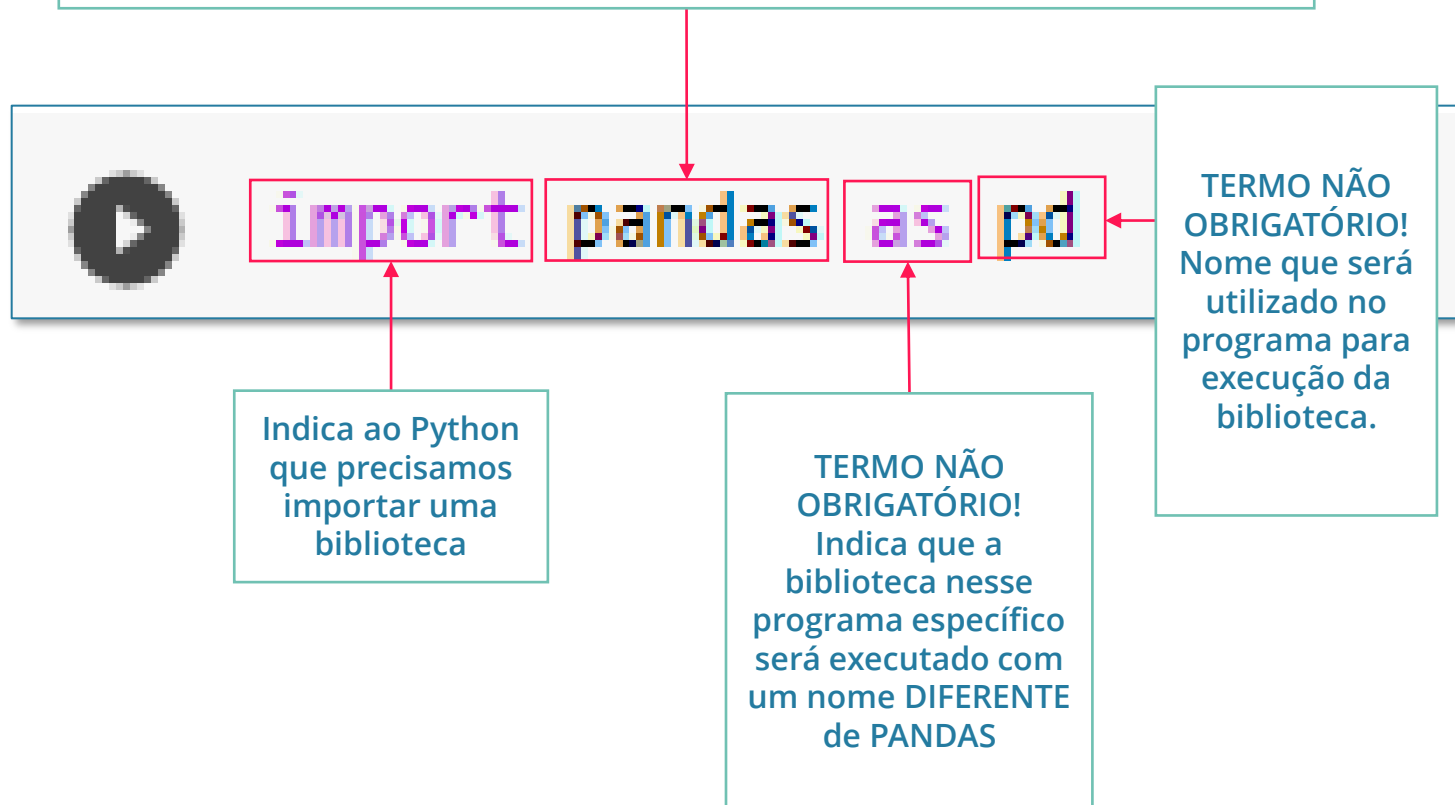
Essas bibliotecas não estão “instaladas” no Python, por isso, precisamos importá-las usando o **IMPORT** como apresentado na figura ao lado.

Para facilitar vamos usar a abreviação **pd**. Ou seja, sempre que quisermos usar o pandas usaremos **pd.COISA QUE QUEREMOS FAZER**.

Vamos abordar isso com um pouco mais de detalhe nas próximas páginas.

Nome da biblioteca. Existem centenas de bibliotecas disponíveis no Python. Só depende da sua curiosidade 😊.

Nesse caso estamos importando a biblioteca PANDAS.
Ela é muito usada no tratamento e análise de grandes quantidades de dados.



Importando base de dados (parte 1/2)

Agora que temos nossa biblioteca **PANDAS** importada, podemos usar uma das suas funções para nos auxiliar a importar a nossa base de dados.

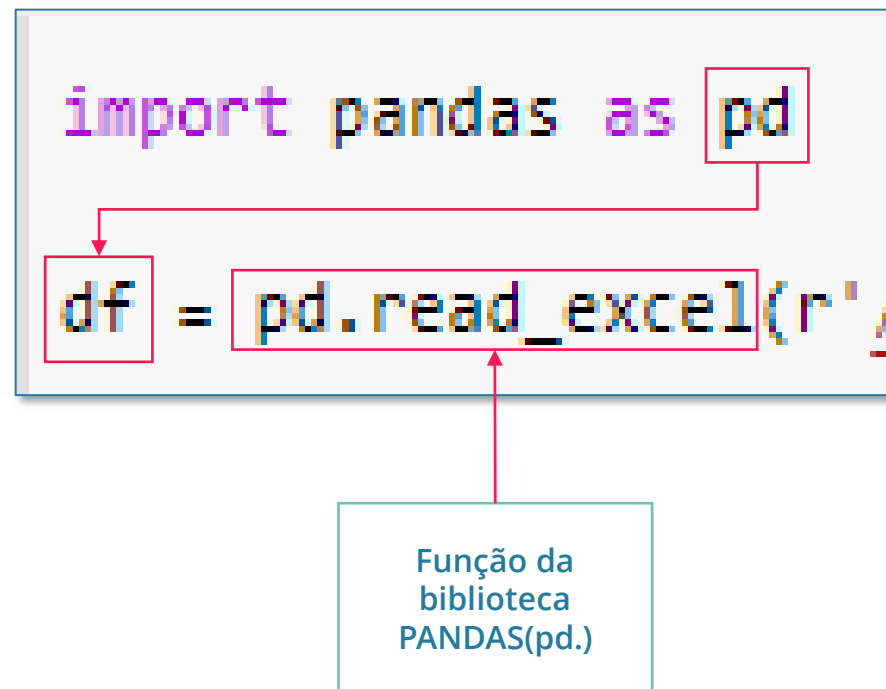
Nesse exemplo, nossa base está no arquivo **Vendas.xls**.

Para acessarmos as fórmulas dentro do **PANDAS** vamos usar a sintaxe:

pd.AÇÃO

Como vimos anteriormente, “**pd**” é uma abreviação para facilitar a programação. Ele representa pandas.

Já ação, nesse caso, será de **Leitura de um arquivo Excel específico**. O nome dessa função no pandas é o “**read_excel**”



Importando base de dados (parte 2/2)

A função **pd.read_excel** necessita de alguns parâmetros para que possa fazer a importação dos dados. São eles:

- Local do arquivo;
- Nome do arquivo;

```
import pandas as pd
```

Caminho no drive onde está salvo o arquivo Vendas.xlsx

```
df = pd.read_excel(r'/content/drive/MyDrive/Colab Notebooks/Projetos Intensivão de Python/Aula 1/Vendas.xlsx')
```

r' indica que o texto é uma *raw string*. Ou seja, sem caracteres especiais.

Nome do arquivo

Além disso, você percebeu que bem no início da linha de código temos “**df =**” ?

No Python, é necessário criarmos um lugar onde seja possível **armazenar os dados** lidos do nosso arquivo excel.

Esse local, chamaremos de **variável**. Existem diversos tipos de variáveis com propósitos distintos.

Por enquanto, o que precisamos entender é que nela, se pode armazenar informação.

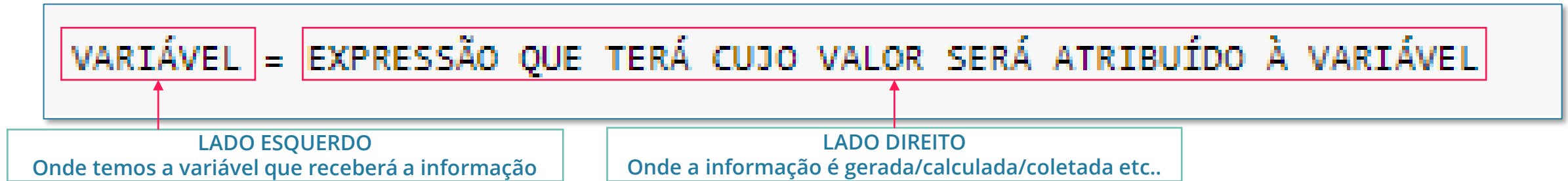
Toda variável possui um nome, no nosso caso, vamos chamá-la de **df** (esse nome é uma abreviação de **dataframe**).

A estrutura ‘**df =**’ deverá ser lida como “**A variável df recebe**”.

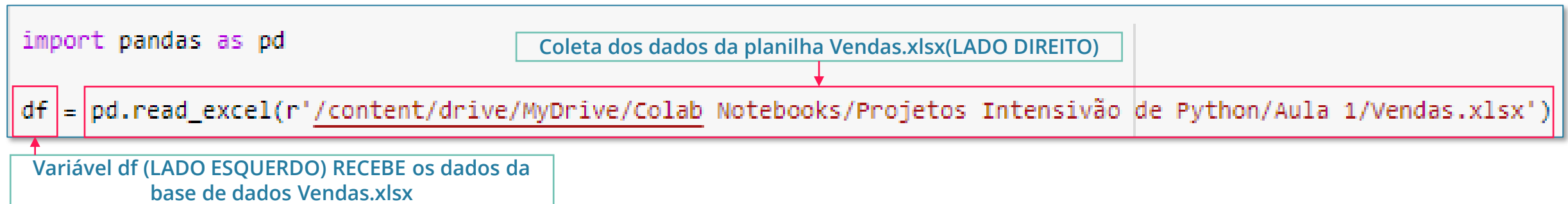
Atribuindo valor a uma variável

Vamos entender um pouco melhor a estrutura que atribui valor a uma variável.

Sempre o que estiver no lado esquerdo do “=” estará recebendo um valor (numérico ou não) da expressão do lado direito.



Agora, vamos analisar novamente nossa linha de código:



Visualizar a base de dados importada

Bem, já importamos nossa base de dados...

Agora vamos tentar visualizá-la !

Usamos a função **DISPLAY()** para exibir nossos dados coletados.

A função display necessita de uma informação: O que será apresentado. Vamos usar nossa variável **df** que está armazenando nossos dados. Assim, temos:

display(df)

Lembrando: para executar o código até aqui clicamos no símbolo de .

Perceba, os dados já foram formatados. Esse é uma das vantagens do **PANDAS**. Ao ler o arquivo excel, os dados já são compilados em uma tabela, o que nos ajuda na visualização.

Além disso, temos informações interessantes como o número de linhas e colunas dessa tabela apresentados na parte inferior da tabela.

```
import pandas as pd

df = pd.read_excel(r'/content/drive/MyDrive/Colab Notebooks/Projetos Intensivo de Python/Aula 1/Vendas.xlsx')

display(df)
```

Função display que apresenta os dados armazenados na variável df

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71
...
100994	69996	2019-12-26	Center Shopping Uberlândia	Short Listrado	2	102	204
100995	69996	2019-12-26	Center Shopping Uberlândia	Mochila	4	270	1080
100996	69996	2019-12-26	Center Shopping Uberlândia	Pulseira Estampa	1	87	87
100997	69997	2019-12-26	Ribeirão Shopping	Camisa Listrado	1	108	108
100998	69997	2019-12-26	Ribeirão Shopping	Short Linho	2	133	266

100999 rows x 7 columns

100999 linhas
7 colunas

Parte 5

Calculando o Faturamento por Loja

Calculando o faturamento por loja

Identificando dados necessários


Agora que temos acesso aos dados, vamos iniciar a resolver nosso desafio.

Vamos começar calculando o faturamento de cada uma das lojas que temos em nossa base de dados.

Para isso, usaremos as colunas **ID LOJA** e **VALOR FINAL** apresentadas na imagem ao lado.

Como temos diversas lojas diferentes usaremos os **métodos***:

- **GROUPBY()** - para agrupar os dados de acordo com alguma referência. Ex: ID Loja;
- **SUM()** - Soma os valores;



Código	Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71
...
100994	69996	2019-12-26	Center Shopping Uberlândia	Short Listrado	2	102	204
100995	69996	2019-12-26	Center Shopping Uberlândia	Mochila	4	270	1080
100996	69996	2019-12-26	Center Shopping Uberlândia	Pulseira Estampa	1	87	87
100997	69997	2019-12-26	Ribeirão Shopping	Camisa Listrado	1	108	108
100998	69997	2019-12-26	Ribeirão Shopping	Short Linho	2	133	266
100999 rows x 7 columns							

Portanto, temos a seguinte linha de código:

```
faturamento = df[['ID Loja', 'Valor Final']].groupby('ID Loja').sum()
```

Variável FATURAMENTO receberá os dados calculados na expressão do lado direito

A estrutura indica que as colunas 'ID Loja' e 'Valor Final' da variável df estão sendo selecionadas.

Soma os dados agrupados

* Métodos são como funções. Por enquanto, não se preocupe em diferenciá-los, apenas entenda que são usados como “ferramentas” que nos ajudam a ser mais rápidos nos códigos

Agrupando e rankeando os dados

Ao agruparmos e somarmos o faturamento de acordo com as lojas existentes, teremos esse resultado.

Mas... Perceba que estes dados estão sem uma ordem específica. Vamos **rankeá-los** considerando a coluna **Valor final** em vendas.

Nesse caso, usaremos outro método:

sort_values()

ID Loja	Valor Final
Bourbon Shopping SP	1726110
Center Shopping Uberlândia	1668921
Iguatemi Campinas	1762419
Iguatemi Esplanada	1699681
Norte Shopping	1711968
Novo Shopping Ribeirão Preto	1678225
Palladium Shopping Curitiba	1721120
Parque Dom Pedro Shopping	1631630
Passei das Águas Shopping	1649014
Ribeirão Shopping	1707122

```
faturamento = faturamento.sort_values(by='Valor Final', ascending=False)
```

Variável FATURAMENTO que usamos anteriormente para receber os valores agrupados por loja

Rankeamento dos valores:

- By = 'Valor Final' -> baseados na coluna Valor Final
- Ascending = False -> Indica que o rankeamento será de forma descendentes

ATENÇÃO! Usaremos a mesma variável faturamento para realizar este rankeamento. (DOIS LADOS DA EXPRESSÃO)

Calculando o faturamento por loja

Visualizando os dados

Usaremos novamente a função `display()` para exibir o resultados até aqui.

```
faturamento = df[['ID Loja', 'Valor Final']].groupby('ID Loja').sum()
faturamento = faturamento.sort_values(by='Valor Final', ascending=False)
display(faturamento)
```

Visualização dos dados

ID Loja	Valor Final
Iguatemi Campinas	1762419
Shopping Vila Velha	1731167
Bourbon Shopping SP	1726110
Rio Mar Recife	1722766
Shopping SP Market	1721763
Palladium Shopping Curitiba	1721120
Norte Shopping	1711968
Ribeirão Shopping	1707122
Iguatemi Esplanada	1699681
Rio Mar Shopping Fortaleza	1698430
Shopping Center Leste Aricanduva	1682870
Novo Shopping Ribeirão Preto	1678225
Shopping Iguatemi Fortaleza	1674824
Center Shopping Uberlândia	1668921

Dados agrupados por Loja e ranqueados do maior para o menor

Parte 5

Calculando a Quantidade Vendida por Loja

Calculando a Quantidade Vendida por Loja

Identificando dados necessários

Assim como fizemos no cálculo do total de vendas, vamos identificar quais são as colunas necessárias para o cálculo da quantidade vendida por loja.

- **ID Loja;**
- **Quantidade**

Usaremos a mesma estrutura usada no passo anterior, no entanto, precisamos criar uma nova variável que receberá esses cálculos.

Chamaremos essa variável de **quantidade**.

Vamos novamente checar como ficarão as linhas de código:



	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71
...
100994	69996	2019-12-26	Center Shopping Uberlândia	Short Listrado	2	102	204
100995	69996	2019-12-26	Center Shopping Uberlândia	Mochila	4	270	1080
100996	69996	2019-12-26	Center Shopping Uberlândia	Pulseira Estampa	1	87	87
100997	69997	2019-12-26	Ribeirão Shopping	Camisa Listrado	1	108	108
100998	69997	2019-12-26	Ribeirão Shopping	Short Linho	2	133	266

100999 rows x 7 columns

```
quantidade = df[['ID Loja', 'Quantidade']].groupby('ID Loja').sum()
quantidade = quantidade.sort_values(by='ID Loja', ascending=False)
display(quantidade)
```

MESMA ESTRUTURA usada no passo anterior. Diferenças: (i)Apenas alteramos as colunas que são utilizadas (ii)Variável quantidade

Calculando a Quantidade Vendida por Loja

Visualizando os dados

Novamente, ranqueamos os dados coletados agrupados por lojas.

As diferença é que agora não estamos usando a coluna **Valor final** e sim a coluna **Quantidade**.

ATENÇÃO! Perceba que não ranqueamos as quantidades, e sim as Lojas.

Como no `sort_values` a opção `ascending=False` foi utilizada, o rankeamento será decrescente.

Ou seja, será o inverso da ordem alfabética.

ID Loja		Quantidade
Shopping Vila Velha		9224
Shopping União de Osasco		8730
Shopping SP Market		8927
Shopping Recife		8581
Shopping Morumbi		8508
Shopping Midway Mall		8206
Shopping Iguatemi Fortaleza		8629
Shopping Ibirapuera		8723
Shopping Eldorado		8719
Shopping Center Leste Aricanduva		8938
Shopping Center Interlagos		8675
Shopping Barra		8638
Salvador Shopping		8698
Rio Mar Shopping Fortaleza		8937

Dados agrupados por Loja e ranqueados de forma decrescente pela coluna ID Loja

Parte 5

Calculando o Ticket Médio dos Produtos por Loja

Calculando o ticket médio dos produtos por loja

Cálculo do ticket médio

Já temos a **variável faturamento** que possui como informação uma base de dados do total de faturamento por loja.

Além disso, temos a **variável quantidade**, que possui como informação uma base de dados do total de itens vendidos por loja.

Se queremos calcular o **ticket médio**, precisamos calcular:

$$\frac{\text{Faturamento por loja}}{\text{Quantidade vendida por loja}}$$

No entanto, temos uma diferença aqui em relação ao passo anterior. Até então, estamos apenas SOMANDO (SUM()) os dados existentes na tabela(df). Aqui, estamos estruturando uma nova coluna e não agrupando uma existente. Para isso, usaremos o método abaixo:

.to_frame()

```
ticket_medio = (faturamento['Valor Final'] / quantidade['Quantidade']).to_frame()
```

Indica que esta variável será uma base de dados (dataframe)

Variável criada para receber a base de dados com o cálculo do ticket médio

Nome da coluna criada.

0

ID Loja	
Bourbon Shopping SP	194.754598
Center Shopping Uberlândia	193.453228
Iguatemi Campinas	197.248909
Iguatemi Esplanada	198.098019
Norte Shopping	189.923231
Novo Shopping Ribeirão Preto	191.775226
Palladium Shopping Curitiba	189.321307
Parque Dom Pedro Shopping	194.519552
Passei das Águas Shopping	191.345324
Ribeirão Shopping	193.441586
Rio Mar Recife	194.377299
Rio Mar Shopping Fortaleza	190.044758
Salvador Shopping	189.323868

Renomeando a coluna da variável ticket_medio

Como vimos anteriormente, a coluna de ticket médio da nossa tabela possui um título pouco intuitivo.

Por default o nome da coluna criada foi '0'.

Vamos usar o método `.rename()` para alterarmos para um nome mais intuitivo e que nos permita melhorar o entendimento da tabela.

Para realizar essa alteração usamos a estrutura:

`columns={0: 'Ticket Médio'}`

Nome atual da coluna

Novo Nome da coluna

Podemos ler este argumento como:

“A coluna que possui valor 0 deverá ser alterado para o nome ‘Ticket Médio’”

```
ticket_medio = (faturamento['Valor Final'] / quantidade['Quantidade']).to_frame()  
ticket_medio = ticket_medio.rename(columns={0: 'Ticket Médio'})
```

Nome inicial da coluna criada → 0	
ID Loja	
Bourbon Shopping SP	194.754598
Center Shopping Uberlândia	193.453228
Iguatemi Campinas	197.248909
Iguatemi Esplanada	198.098019
Norte Shopping	189.923231
Novo Shopping Ribeirão Preto	191.775226
Palladium Shopping Curitiba	189.321307
Parque Dom Pedro Shopping	194.519552
Passei das Águas Shopping	191.345324
Ribeirão Shopping	193.441586
Rio Mar Recife	194.377299
Rio Mar Shopping Fortaleza	190.044758
Salvador Shopping	189.323868

Ticket Médio	
ID Loja	
Bourbon Shopping SP	194.754598
Center Shopping Uberlândia	193.453228
Iguatemi Campinas	197.248909
Iguatemi Esplanada	198.098019
Norte Shopping	189.923231
Novo Shopping Ribeirão Preto	191.775226
Palladium Shopping Curitiba	189.321307
Parque Dom Pedro Shopping	194.519552
Passei das Águas Shopping	191.345324
Ribeirão Shopping	193.441586
Rio Mar Recife	194.377299
Rio Mar Shopping Fortaleza	190.044758
Salvador Shopping	189.323868

Ordenando a coluna e visualizando os dados

Assim como fizemos anteriormente, vamos ranquear os dados, do maior para o menor, baseado na coluna **Ticket Medio**.

Além disso, vamos novamente visualizar os dados obtidos.

```
ticket_medio = (faturamento['Valor Final'] / quantidade['Quantidade']).to_frame()  
ticket_medio = ticket_medio.rename(columns={0: 'Ticket Medio'})  
ticket_medio = ticket_medio.sort_values(by='Ticket Medio', ascending=False)  
display(ticket_medio)
```

Ticket Medio	
ID Loja	
Iguatemi Esplanada	198.098019
Iguatemi Campinas	197.248909
Bourbon Shopping SP	194.754598
Parque Dom Pedro Shopping	194.519552
Rio Mar Recife	194.377299
Shopping Iguatemi Fortaleza	194.092479
Shopping Midway Mall	193.814404
Center Shopping Uberlândia	193.453228
Ribeirão Shopping	193.441586
Shopping SP Market	192.871401
Novo Shopping Ribeirão Preto	191.775226
Shopping Barra	191.375666

Parte 6

Criando a função de enviar e-mail

Revisando o que fizemos antes de continuar

Vamos revisar o que fizemos até agora e que informações temos disponíveis.

- 1) Obtemos os dados da base Vendas.xlsx;
- 2) Organizamos os dados usando o PANDAS;
- 3) Extraímos da base de dados original **df, 3 tabelas específicas**:
 - 1) **Faturamento por loja;**
 - 2) **Quantidade vendida por loja;**
 - 3) **Ticket médio por loja.**

- 4) Ordenamos todos os dados, do maior para o menor, para facilitar a visualização;

Agora que temos os dados, vamos para parte do relatório. Nosso objetivo é que cada um dos gerentes de loja receba, no e-mail, um relatório com as informações mais importantes da sua loja.

Para isso, vamos precisar criar uma função que:

- 1) Colete as informações que geramos;
- 2) Crie o e-mail com os dados da loja;
- 3) Envie o e-mail para o gerente daquela loja específica;
- 4) Enviar e-mail para Diretoria.



Criando a função de enviar e-mail

Configurações Importantes

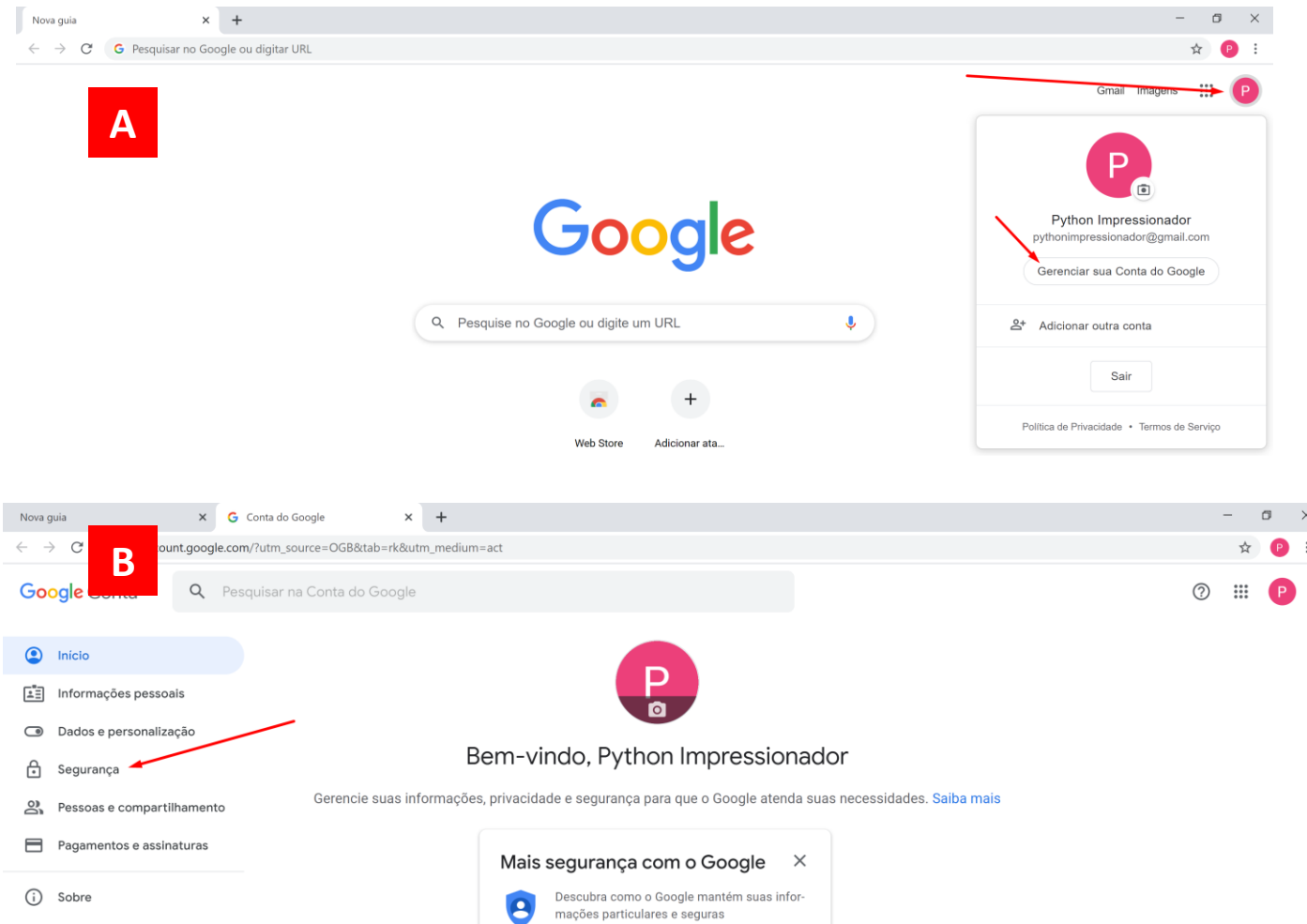
Vamos usar uma função que envia e-mail pelo gmail.

Para isso, precisamos habilitar o seu gmail para enviar mensagens por código. Isso porque o gmail bloqueia automaticamente esses códigos por precaução, mas podemos liberar. Apenas 2 passos são necessários para isso.

Passo 1: Habilitar Aplicativos Menos Seguros

Aqui, você vai entrar na sua conta de e-mail e seguir os seguintes passos:

- A: Gerenciar sua Conta do Google
- B: Segurança
- C: Acesso a app menos seguro
- D: Ativar



Criando a função de enviar e-mail

Configurações Importantes

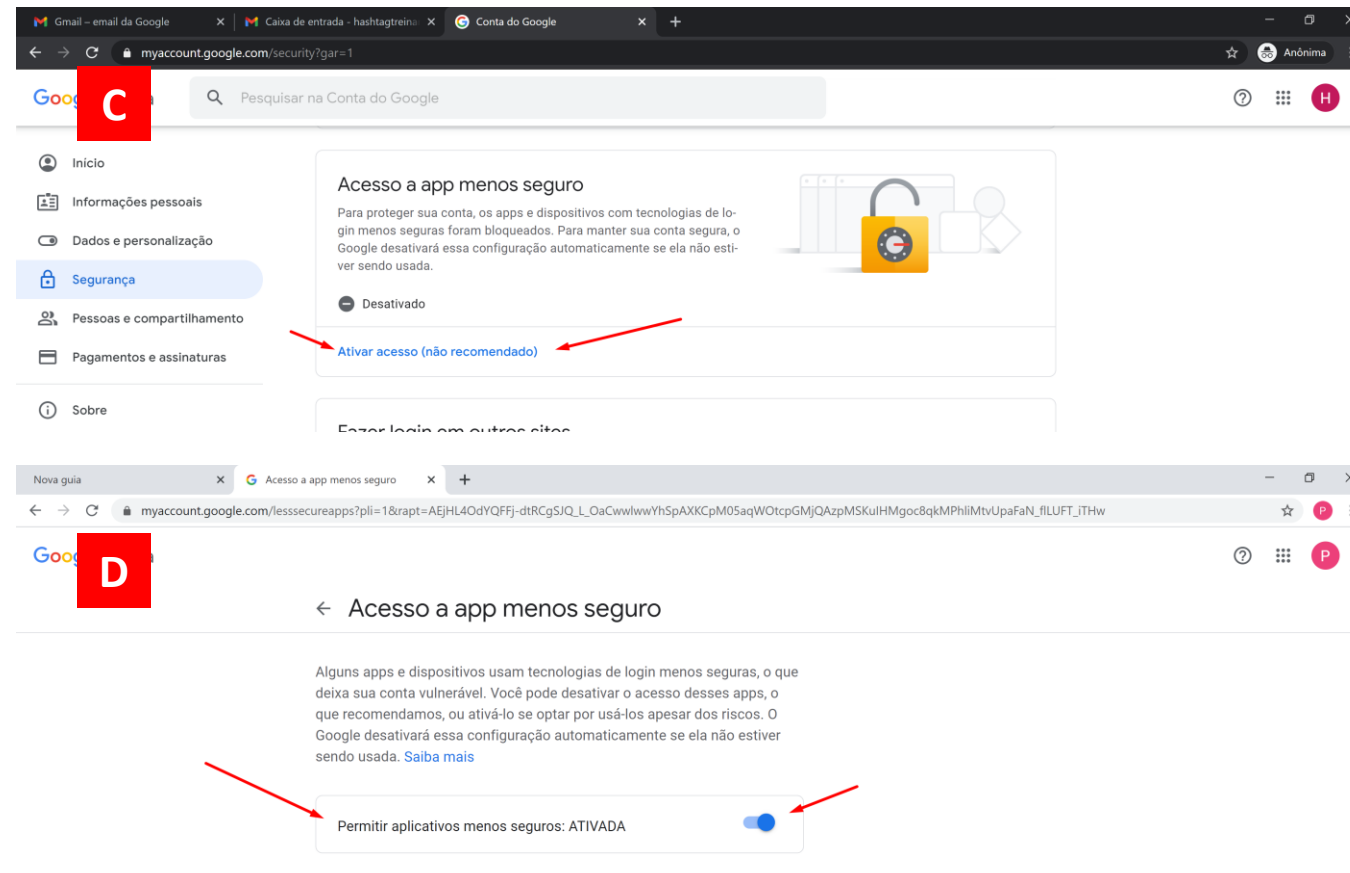
Vamos usar uma função que envia e-mail pelo gmail.

Para isso, precisamos habilitar o seu gmail para enviar mensagens por código. Isso porque o gmail bloqueia automaticamente esses códigos por precaução, mas podemos liberar. Apenas 2 passos são necessários para isso.

Passo 1: Habilitar Aplicativos Menos Seguros

Aqui, você vai entrar na sua conta de e-mail e seguir os seguintes passos:

- A: Gerenciar sua Conta do Google
- B: Segurança
- C: Acesso a app menos seguro
- D: Ativar



Criando a função de enviar e-mail

Configurações Importantes

Passo 2: Desabilitar Bloqueio de Apps

Aqui, você só precisa entrar em um link e habilitar o desbloqueio

Link para acessar:

<https://accounts.google.com/DisplayUnlockCaptcha>

Depois disso, basta clicar em “Continuar”



Criando função de envio de e-mails (1/7)

Criar uma função no Python, significa, a grosso modo, criar um programa dentro de um programa. Imagine que você sempre precisa enviar um e-mail específico, mas apenas 2 itens mudam nesse e-mail. Ao invés de sempre alterar o seu código, você pode criar uma função que se chama MANDAR EMAIL(ITEM 1 , ITEM 2), onde, item 1 e item 2, são as duas coisas que sempre variam.

Isso nos dá mais velocidade na elaboração de programas.

Para criar funções, usamos o termo **def** seguido do nome da função que desejamos criar. Geralmente esse nome é um verbo no infinitivo, pois estamos executando uma ação. Ex: Calcular, Somar, Contar, etc...

Além disso, para esse caso de envio de e-mails vamos usar outra biblioteca disponível no Python que nos ajudará muito na elaboração da função: o **smtplib** e **email.message**.

```
# função enviar_email
```

```
import smtplib
```

```
import email.message
```

Importação de bibliotecas

```
def enviar_email(resumo_loja, loja):
```

Indica a criação de uma função

Nome da função a ser criada. Perceba que usamos um verbo no infinitivo

Argumentos da função. Estes argumentos serão necessários para execução da função. Variáveis `resumo_loja` e `loja` ainda não foram criadas

Criando função de envio de e-mails (2/7)

O que vamos fazer agora, vai parecer um tanto quanto estranho, mas é muito comum na programação em Python.

Por se tratar de um código aberto, as bibliotecas nos ajudam muito a acelerar o processo.

Elas foram criadas exatamente para que você não precisa criá-las. Portanto, é mais uma questão de entender como ela funciona e preencher os dados que são particulares do seu projeto do que de fato programar do zero.

Acredite ou não, a função ao lado é essencialmente extraída da internet. O que temos que mudar? O que for particular a nossa realidade.

Vamos analisar cada um dos blocos de código para entendermos o que é a biblioteca e o que é particular.

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja):
    server = smtplib.SMTP('smtp.gmail.com:587')
    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = 'joaoprira@poli.ufrj.br'
    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)

    s = smtplib.SMTP('smtp.gmail.com: 587')
    s.starttls()
    # Login Credentials for sending the mail
    s.login(msg['From'], password)
    s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```

Esse espaço é chamado de indentação.

Esse espaço é o que indica para o Python que todas as linhas de código que a possuem estão “dentro” do bloco def de criação da função

Criando função de envio de e-mails (3/7)

Antes de analisarmos os blocos de código, precisamos lembrar que não estamos escrevendo um código específico para envio dos e-mails das lojas, e sim, uma função **GENÉRICA** de envio de e-mails que precisam de 2 variáveis:

- `resumo_loja`;
- `loja`.

Voltando para o bloco de código indicado, podemos ver que foi criada uma variável **server**. Esta variável irá indicar que tipo de servidor estamos falando.

Nesse caso, usaremos um GMAIL, por isso, usamos este argumento dentro do parênteses. Isso é algo descrito na documentação da biblioteca `smtplib` e não precisa ser “decorada”.

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja):

    server = smtplib.SMTP('smtp.gmail.com:587')
    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = 'joaoprirlira@poli.ufrj.br'
    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)

    s = smtplib.SMTP('smtp.gmail.com: 587')
    s.starttls()
    # Login Credentials for sending the mail
    s.login(msg['From'], password)
    s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```

Criando função de envio de e-mails (4/7)

Seguindo o bloco de código, temos a criação de uma variável **email_content** que receberá o conteúdo deste e-mail.

Podemos dividir este conteúdo em 4 partes:

- 1) `f''`:
 - Indica que não temos apenas um texto e sim um texto com alguma variável integrada;
- 2) `<p> Coe Lira,<p>`:
 - `<p>`: Indica que teremos um parágrafo
- 3) `{resumo_loja.to_html()}:`
 - Indica que o valor atribuída à variável `resumo_loja`, será inserida nesta parte do conteúdo do texto.

OBS: HTML significa o formato de programação a ser utilizado.
- 4) `<p> Tmj<p>`:
 - `<p>` indica que teremos um parágrafo

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja):

    server = smtplib.SMTP('smtp.gmail.com:587')

    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = 'joaoprirlira@poli.ufrj.br'
    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)

    s = smtplib.SMTP('smtp.gmail.com: 587')
    s.starttls()
    # Login Credentials for sending the mail
    s.login(msg['From'], password)
    s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```

Criando função de envio de e-mails (5/7)

No segundo bloco da função temos duas linhas de código.

Na primeira criamos mais uma variável `msg` que receberá as informações existentes em um e-mail. Perceba que utilizamos a biblioteca `email.message` para realizar essa operação.

A segunda linha `msg['Subject']` nos permite criar um título padrão para o assunto onde apenas será variável a `{loja}` que como já vimos, deverá ser fornecida pelo usuário.

Da mesma forma usaremos o `msg`, para criarmos o remetente (`['From']`) e o destinatário (`['To']`) do e-mail.

Obs: Aqui você deverá trocar para seu e-mail para que funcione no seu pc.

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja):


    server = smtplib.SMTP('smtp.gmail.com:587')
    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = 'joaoprllira@poli.ufrj.br'

    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)

    s = smtplib.SMTP('smtp.gmail.com: 587')
    s.starttls()
    # Login Credentials for sending the mail
    s.login(msg['From'], password)
    s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```



Criando função de envio de e-mails (6/7)

Continuando nossa linhas de código, temos a criação de uma variável **password**. Aqui, obviamente, não colocamos a senha do e-mail do Lira, mas em teoria, você pode colocar sua **senha**.

ATENÇÃO! O ideal, é que quando tratamos de senhas em códigos, é que o usuário tenha um momento de inseri-las manualmente.

Afinal, imagina se alguém mais consegue acessar seu código... Verá sua senha, e você pode estar tendo um risco de segurança!!!

Portanto, aqui usaremos a senha, mas tenha em mente que isso pode ser uma vulnerabilidade do seu código.

Dito isso, vamos continuar...

Além da senha, vamos também entender os métodos usados nas linhas seguintes:

- **.add_header()**: inclui um cabeçalho ao e-mail;
- **.set_payload(e-mail_content)**: Inclui em nossa mensagem o conteúdo que definimos anteriormente na variável **e-mail_content**.

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja):

    server = smtplib.SMTP('smtp.gmail.com:587')

    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = 'joaoprira@poli.ufrj.br'

    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)

    s = smtplib.SMTP('smtp.gmail.com: 587')
    s.starttls()
    # Login Credentials for sending the mail
    s.login(msg['From'], password)
    s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```

Criando função de envio de e-mails (7/7)

No último bloco iremos usar novamente a biblioteca **smtplib** para:

- 1) inicializarmos o gmail;
- 2) ingressarmos no nosso e-mail;
- 3) Enviar o e-mail.

LEMBRANDO!!!

Até agora apenas criamos a função que envia e-mails.

Precisamos agora, criar um código que nos permita interpretar os dados coletados até aqui e aplica-los nessa função **enviar_email** que acabamos de criar.

Vamos lá!

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja):

    server = smtplib.SMTP('smtp.gmail.com:587')
    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = 'joaoprirlira@poli.ufrj.br'
    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)
```

```
s = smtplib.SMTP('smtp.gmail.com: 587')
s.starttls()
# Login Credentials for sending the mail
s.login(msg['From'], password)
s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```

Parte 7

Criando o relatório por loja

Criando o relatório por loja

Criando a variável loja

Primeiramente, vamos criar nossa variável **loja** que será necessária na nossa função **enviar_email**.

Para isso, vamos usar a coluna **'ID loja'** do nosso *dataframe* **df**.

Como podemos ver, essa coluna possui valores repetidos, portanto, vamos usar um método que nos permite ter uma lista de valores únicos. Esse método é o^:

.unique().

```
lojas = df['ID Loja'].unique()
```

ID Loja
Iguatemi Esplanada
Iguatemi Esplanada
Iguatemi Esplanada
Norte Shopping
Norte Shopping
...
Center Shopping Uberlândia
Center Shopping Uberlândia
Center Shopping Uberlândia
Ribeirão Shopping
Ribeirão Shopping

```
['Iguatemi Esplanada', 'Norte Shopping',  
'Rio Mar Shopping Fortaleza', 'Shopping Barra',  
'Shopping Ibirapuera', 'Iguatemi Campinas',  
'Shopping Center Leste Aricanduva', 'Passei das Águas Shopping',  
'Shopping Recife', 'Shopping Midway Mall', 'Bourbon Shopping SP',  
'Shopping Center Interlagos', 'Parque Dom Pedro Shopping',  
'Center Shopping Uberlândia', 'Shopping União de Osasco',  
'Shopping Eldorado', 'Shopping Vila Velha',  
'Novo Shopping Ribeirão Preto', 'Rio Mar Recife',  
'Palladium Shopping Curitiba', 'Shopping SP Market',  
'Ribeirão Shopping', 'Shopping Iguatemi Fortaleza',  
'Shopping Morumbi', 'Salvador Shopping'], dtype=object)
```

25 lojas
distintas

Criando o relatório por loja

Como temos 25 lojas distintas, escrever um código específico para cada uma delas não parece ser a forma mais prática e eficiente de se agir.

Imagine que ao invés de 25 fossem 500? Não parece prático né?

Para isso, temos **estruturas de repetição** no Python. A mais comum delas é o **FOR**. Ela nos permite percorrer todas as nossas 25 lojas e criar um relatório de forma automática.

Aqui usaremos a **indentação** novamente. Logo, todas as linhas que estão indentadas pertencem ao bloco **for**.

A estrutura **loja in lojas**, significa que uma loja por vez da lista lojas irá passar pelas 3 linhas pertencentes ao for.

Abaixo do código, fazemos duas iterações teóricas para que fique mais claro.

```
lojas = df['ID Loja'].unique()

for loja in lojas:
    tabela_loja = df.loc[df['ID Loja'] == loja, ['ID Loja', 'Quantidade', 'Valor Final']]
    resumo_loja = tabela_loja.groupby('ID Loja').sum()
    resumo_loja['Ticket Médio'] = resumo_loja['Valor Final'] / resumo_loja['Quantidade']
```

1º Iteração

Lojas:

['Iguatemi Esplanada', 'Norte Shopping',
'Rio Mar Shopping Fortaleza', 'Shopping Barra',

```
tabela_loja = df.loc[df['ID Loja'] == 'Iguatemi Esplanada', ['ID Loja', 'Quantidade', 'Valor Final']]
resumo_loja = tabela_loja.groupby('ID Loja').sum()
Resumo_loja['Ticket Médio'] = resumo_loja['Valor Final'] / resumo_loja['Quantidade']
```

2º Iteração

Lojas:

['Iguatemi Esplanada', 'Norte Shopping',
'Rio Mar Shopping Fortaleza', 'Shopping Barra',

```
tabela_loja = df.loc[df['ID Loja'] == 'Norte Shopping', ['ID Loja', 'Quantidade', 'Valor Final']]
resumo_loja = tabela_loja.groupby('ID Loja').sum()
Resumo_loja['Ticket Médio'] = resumo_loja['Valor Final'] / resumo_loja['Quantidade']
```


Parte 8

Enviando o e-mail automaticamente

Enviando o e-mail por loja

Podemos aproveitar as iterações para já enviarmos o e-mail da loja .

Para isso, vamos usar nossa função **enviar_email**!

Vamos lembrar quais são seus argumentos:


Enviar_email(resumo_loja , loja)

Podemos perceber que `resumo_loja` é calculado dentro do FOR.

Já `loja`, como vimos no slide anterior, irá variar até que o Python passe por todas as lojas.

```
lojas = df['ID Loja'].unique()

for loja in lojas:
    tabela_loja = df.loc[df['ID Loja'] == loja, ['ID Loja', 'Quantidade', 'Valor Final']]
    resumo_loja = tabela_loja.groupby('ID Loja').sum()
    resumo_loja['Ticket Médio'] = resumo_loja['Valor Final'] / resumo_loja['Quantidade']
    enviar_email(resumo_loja, loja)
```



ID Loja	Quantidade	Valor Final	Ticket Médio
Iguatemi Esplanada	8580	1699681	198.098019

Enviando o e-mail para a diretoria

Para a diretoria iremos seguir o mesmo conceito, mas por se tratar de um nível hierárquico maior, vamos agrupar toda a informação em uma única tabela para que seja possível uma visão macro de como vão os resultados da empresa.


Para isso, usaremos nossas variáveis iniciais:

- faturamento;
- quantidade;
- ticket_médio;

Para unirmos as três informações usaremos o método `.join()` conforme apresentado ao lado.

```
# email para diretoria
```

```
tabela_diretoria = faturamento.join(quantidade).join(ticket_medio)  
enviar_email(tabela_diretoria, 'Todas as Lojas')
```



ID Loja	Valor Final	Quantidade	Ticket Medio
Iguatemi Campinas	1762419	8935	197.248909
Shopping Vila Velha	1731167	9224	187.680724
Bourbon Shopping SP	1726110	8863	194.754598
Rio Mar Recife	1722766	8863	194.377299
Shopping SP Market	1721763	8927	192.871401
Palladium Shopping Curitiba	1721120	9091	189.321307
Norte Shopping	1711968	9014	189.923231
Ribeirão Shopping	1707122	8825	193.441586
Iguatemi Esplanada	1699681	8580	198.098019
Rio Mar Shopping Fortaleza	1698430	8937	190.044758
Shopping Center Leste Aricanduva	1682870	8938	188.282614
Novo Shopping Ribeirão Preto	1678225	8751	191.775226

Parte 9

Checando os resultados

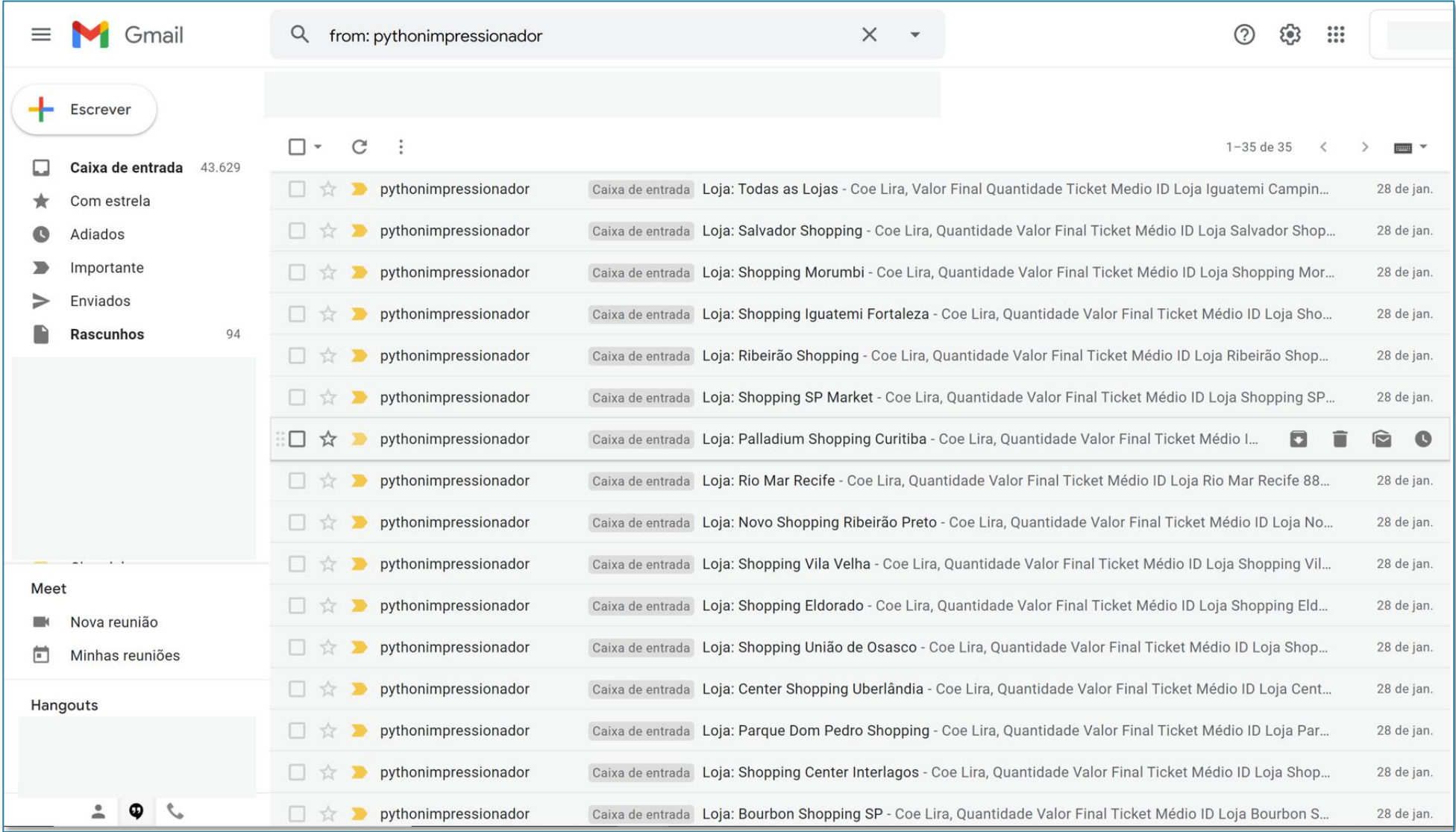
Checando os resultados

Caixa de entrada

Pronto!

Todos os e-mails enviados em poucos segundos 😊.

Só o início para se tornar um **IMPRESSIONADOR**.



E se eu quisesse enviar e-mail para 25 e-mails diferentes?

Nesse caso, no nosso arquivo em Excel precisaria ter uma coluna de e-mails, para a gente poder fazer isso.

Digamos então que esse fosse o caso. Nossa base de dados seria assim:

Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final	Email
1	01/01/2019	Iguatemi Esplanada	Sapato Estampa	1	R\$ 358,00	R\$ 358,00	iguatemiesplanada@gmail.com
1	01/01/2019	Iguatemi Esplanada	Camiseta	2	R\$ 180,00	R\$ 360,00	iguatemiesplanada@gmail.com
1	01/01/2019	Iguatemi Esplanada	Sapato Xadrez	1	R\$ 368,00	R\$ 368,00	iguatemiesplanada@gmail.com
2	02/01/2019	Norte Shopping	Relógio	3	R\$ 200,00	R\$ 600,00	norteshopping@gmail.com
2	02/01/2019	Norte Shopping	Chinelo Liso	1	R\$ 71,00	R\$ 71,00	norteshopping@gmail.com
3	02/01/2019	Rio Mar Shopping Fortaleza	Cinto Linho	1	R\$ 248,00	R\$ 248,00	riomarshopping@gmail.com
5	02/01/2019	Shopping Barra	Calça	1	R\$ 170,00	R\$ 170,00	shoppingbarra@gmail.com
6	02/01/2019	Shopping Ibirapuera	Polo Listrado	4	R\$ 149,00	R\$ 596,00	shoppingibirapuera@gmail.com
7	02/01/2019	Norte Shopping	Camisa Gola V Listrado	1	R\$ 116,00	R\$ 116,00	norteshopping@gmail.com
7	02/01/2019	Norte Shopping	Camisa Liso	1	R\$ 105,00	R\$ 105,00	norteshopping@gmail.com
8	02/01/2019	Iguatemi Campinas	Terno Estampa	5	R\$ 706,00	R\$ 3.530,00	iguaticampinas@gmail.com
9	02/01/2019	Shopping Center Leste Aricanduva	Tênis Linho	1	R\$ 294,00	R\$ 294,00	shoppingcenterlestearicanduva@gmail.com

Em seguida, precisaríamos criar uma lista com os e-mails, igual fizemos com as lojas:

```
lista_emails = df['Email'].unique()
```

O resultado seria uma lista com todos os e-mails. Em que cada e-mail aparece uma única vez.

E se eu quisesse enviar e-mail para 25 e-mails diferentes?

Feito isso, precisamos agora modificar o nosso “for” para poder rastrear o e-mail correspondente de cada loja.

Para isso, vamos modificar a nossa linha de código do “for” para:

`for i, loja in enumerate(lojas):`

Isso vai fazer com que o `i` seja o índice da loja que estamos pegando. Precisamos desse índice porque o 1º e-mail corresponde a primeira loja. O 2º e-mail corresponde a segunda loja e assim vai. O índice cria essa correspondência. Conforme a imagem ao lado

Então como a nossa lista de e-mails se chama `lista_emails`, sempre que escrevermos dentro do “for” o código `lista_emails[i]` estamos pegando o e-mail correspondente a loja.

Então agora, só precisamos passar essa informação para a nossa função `enviar_email`

Afinal de contas, ela precisa saber para qual e-mail deve enviar a resposta.

	ID Loja	Email
0	Iguatemi Esplanada	iguatemiesplanada@gmail.com
1	Norte Shopping	norteshopping@gmail.com
2	Rio Mar Shopping Fortaleza	riomarshopping@gmail.com
3	Shopping Barra	shoppingbarra@gmail.com
4	Shopping Ibirapuera	shoppingibirapuera@gmail.com
5	Iguatemi Campinas	iguaticampinas@gmail.com
6	Shopping Center Leste Aricanduva	shoppingcenterlestearicanduva@gmail.com
7	Passei das Águas Shopping	passeidasaguasshopping@gmail.com
8	Shopping Recife	shoppingrecife@gmail.com

E se eu quisesse enviar e-mail para 25 e-mails diferentes?

Com isso, o bloco de código que contém o for ficou assim:

```
▶ lojas = df['ID Loja'].unique()
lista_emails = df['ID Loja'].unique()

for i, loja in enumerate(lojas):
    tabela_loja = df.loc[df['ID Loja'] == loja, ['ID Loja', 'Quantidade', 'Valor Final']]
    resumo_loja = tabela_loja.groupby('ID Loja').sum()
    resumo_loja['Ticket Médio'] = resumo_loja['Valor Final'] / resumo_loja['Quantidade']
    enviar_email(resumo_loja, loja, lista_emails[i])
```

E a função de enviar_email também precisa ser editada, para poder receber 3 informações agora e atualizar o destinatário a cada novo e-mail.

São apenas 2 passos para isso:

Passo 1: Acrescentar um novo parâmetro na linha “def enviar_email(resumo_loja, loja, email) -> Acrescentamos o parâmetro email

Passo 2: Na linha de código do destinatário, ao invés de colocarmos um destinatário fixo, adaptamos o destinatário para o parâmetro email

E se eu quisesse enviar e-mail para 25 e-mails diferentes?

Com isso, o bloco de código da função `enviar_email` fica assim:

Pronto.

Lembrando, precisamos executar primeiro o bloco de código da função `enviar_email` (para o Python atualizar a função `enviar_email` com as novas modificações)

E só depois executar o código para disparar os e-mails para cada loja.

```
# função enviar_email
import smtplib
import email.message

def enviar_email(resumo_loja, loja, email):

    server = smtplib.SMTP('smtp.gmail.com:587')
    email_content = f'''
    <p>Coe Lira,</p>
    {resumo_loja.to_html()}
    <p>Tmj</p>'''

    msg = email.message.Message()
    msg['Subject'] = f'Lira Rules - Loja: {loja}'

    msg['From'] = 'pythonimpressionador@gmail.com'
    msg['To'] = email
    password = senha
    msg.add_header('Content-Type', 'text/html')
    msg.set_payload(email_content)

    s = smtplib.SMTP('smtp.gmail.com: 587')
    s.starttls()
    # Login Credentials for sending the mail
    s.login(msg['From'], password)
    s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
```

INTENSIVÃO DE PYTHON {#}

100% ONLINE & GRATUITO

Ainda não segue a gente no Instagram e nem é inscrito no nosso canal do Youtube? Então corre lá!



@hashtagprogramacao



youtube.com/hashtag-programacao

