



{JORNADA}  
PYTHON

AULA 1

# PYTHON POWER UP



Parte 1

# Introdução

# O que Vamos Aprender?

Na aula **Python Power UP** nós vamos fazer um **projeto de automação**. Imagine que você tenha produtos, preços ou qualquer tipo de informação que precisa cadastrar em um sistema.

Agora imagine que você tem 500 informações, seria um tanto trabalhoso ter que fazer tudo isso de forma manual não é mesmo?

Você perderia até alguns dias de trabalho fazendo esse processo. Só que nessa aula, eu vou te mostrar como você pode automatizar um processo desses utilizando o Python.

**Você vai aprender como controlar o mouse e o teclado** para que executem tarefas como se fosse você! É exatamente isso mesmo que você leu, o Python vai fazer isso por você.

A ideia do projeto é que você consiga entrar em um site (site criado especialmente para essa aula), fazer seu login e fazer o cadastro de todos os produtos de forma automática.

Lembrando que vamos ter uma base de dados com esses produtos para que você possa simular sua lista de produtos, preços ou qualquer outra informação que tenha para registrar

### Sistema Python PowerUp

Faça seu Login para acessar

**E-mail**

**Senha**

Logar

### Formulário de Cadastro de Produtos

**Código do Produto**

**Marca do Produto**

**Tipo do Produto**

**Categoria do Produto**

**Preço Unitário do Produto**

**Custo do Produto**

**OBS**

Enviar Limpar

# Entendendo a Base de Dados

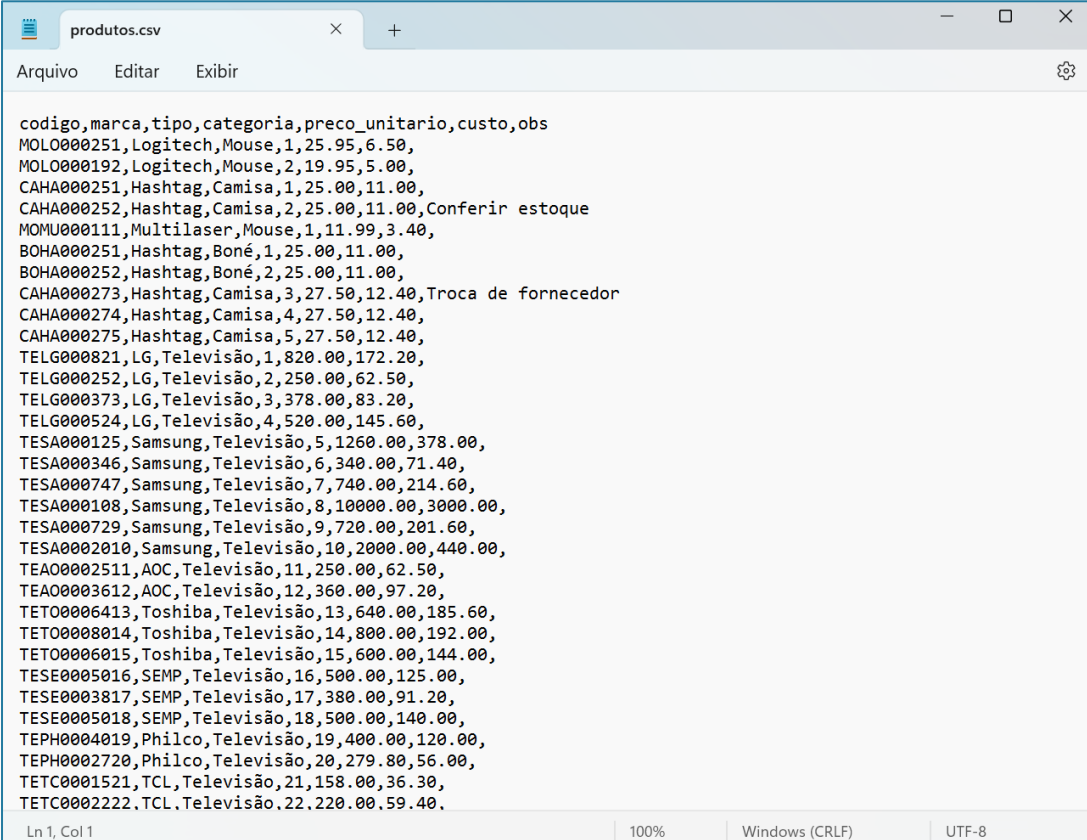
Os dados que nós vamos utilizar são as informações que estão dentro do arquivo **produtos.csv**. Nesse arquivo temos diversos produtos para serem cadastrados no sistema de forma automática.

Você provavelmente vai ter uma lista dos itens que precisa cadastrar quando for fazer um procedimento similar não é mesmo? Então aqui temos exatamente essa lista (base de dados) com as informações a serem cadastradas.

Temos o código do produto, marca, tipo, preço unitário, custo e observação. Só que o primeiro ponto, é que as informações estão no formato **csv**, o que dificulta um pouco a visualização.

Outro ponto é que temos **264 linhas** de informação, então você teria que preencher **6 informações no sistema para cada linha** da base de dados.

É para evitar esse tipo de trabalho manual que vamos criar automações com Python. Com isso o trabalho fica muito mais rápido, automático e sem chances de inserir informações erradas.



```
codigo,marca,tipo,categoria,preco_unitario,custo,obs
MOL0000251,Logitech,Mouse,1,25.95,6.50,
MOL0000192,Logitech,Mouse,2,19.95,5.00,
CAHA000251,Hashtag,Camisa,1,25.00,11.00,
CAHA000252,Hashtag,Camisa,2,25.00,11.00,Conferir estoque
MOMU000111,Multilaser,Mouse,1,11.99,3.40,
BOHA000251,Hashtag,Boné,1,25.00,11.00,
BOHA000252,Hashtag,Boné,2,25.00,11.00,
CAHA000273,Hashtag,Camisa,3,27.50,12.40,Troca de fornecedor
CAHA000274,Hashtag,Camisa,4,27.50,12.40,
CAHA000275,Hashtag,Camisa,5,27.50,12.40,
TELG000821,LG,Televisão,1,820.00,172.20,
TELG000252,LG,Televisão,2,250.00,62.50,
TELG000373,LG,Televisão,3,378.00,83.20,
TELG000524,LG,Televisão,4,520.00,145.60,
TESA000125,Samsung,Televisão,5,1260.00,378.00,
TESA000346,Samsung,Televisão,6,340.00,71.40,
TESA000747,Samsung,Televisão,7,740.00,214.60,
TESA000108,Samsung,Televisão,8,10000.00,3000.00,
TESA000729,Samsung,Televisão,9,720.00,201.60,
TESA0002010,Samsung,Televisão,10,2000.00,440.00,
TEAO0002511,AOC,Televisão,11,250.00,62.50,
TEAO0003612,AOC,Televisão,12,360.00,97.20,
TETO0006413,Toshiba,Televisão,13,640.00,185.60,
TETO0008014,Toshiba,Televisão,14,800.00,192.00,
TETO0006015,Toshiba,Televisão,15,600.00,144.00,
TESE0005016,SEMP,Televisão,16,500.00,125.00,
TESE0003817,SEMP,Televisão,17,380.00,91.20,
TESE0005018,SEMP,Televisão,18,500.00,140.00,
TEPH0004019,Philco,Televisão,19,400.00,120.00,
TEPH0002720,Philco,Televisão,20,279.80,56.00,
TETC0001521,TCL,Televisão,21,158.00,36.30,
TETC0002222,TCL,Televisão,22,220.00,59.40,
```

# Introdução

## Entendendo a Solução Final

A solução desse projeto vai ser exatamente a nossa automação, ou seja, vamos ter no final todos os produtos da nossa base de dados cadastrados no sistema de forma automática, como se você estivesse utilizando o computador para fazer o cadastro.

Ao rodar o programa vamos ter as seguintes ações para completar o nosso cadastro:

- Abrir o navegador
- Acessar o site do sistema com login e senha
- Inserir todas as informações do produto
- Enviar as informações para o sistema
- Repetir o cadastro até acabar o cadastro de todos os produtos

Sistema Python PowerUp

Faça seu Login para acessar

E-mail

pythonimpressionador@gmail.com

Senha

\*\*\*\*\*

Logar

Formulário de Cadastro de Produtos

Código do Produto

CAHA000273

Marca do Produto

Hashtag

Tipo do Produto

Camisa

Categoria do Produto

3

Preço Unitário do Produto

27.5

Custo do Produto

12.4

OBS

Troca de fornecedor

Enviar

Limpar

Produtos Cadastrados						
Código	Marca	Tipo	Categoria	Preço Un.	Custo	OBS
MOLO000251	Logitech	Mouse	1	25.95	6.5	
MOLO000192	Logitech	Mouse	2	19.95	5.0	
CAHA000251	Hashtag	Camisa	1	25.0	11.0	
CAHA000252	Hashtag	Camisa	2	25.0	11.0	Conferir estoque
MOMU000111	Multilaser	Mouse	1	11.99	3.4	
BOHA000251	Hashtag	Bon	1	25.0	11.0	
BOHA000252	Hashtag	Bon	2	25.0	11.0	
CAHA000273	Hashtag	Camisa	3	27.5	12.4	Troca de fornecedor

Parte 2

# Importando e Visualizando os Dados



# Importando Base de Dados

Assim como na primeira aula da Jornada Python, nós vamos utilizar a biblioteca pandas para importar a base de dados e podemos visualizá-la para que você entenda como os dados estão organizados.

A base de dados é o arquivo **produtos.csv**, então é bom ter o arquivo no mesmo local em que criou o seu arquivo em Python.

Dessa forma pode só utilizar o comando **pd.read\_csv** e colocar o nome do arquivo (caso contrário vai ter que colocar o caminho completo do arquivo).

Depois basta utilizar o print para visualizar os dados.

```
1 import pyautogui
2 import pandas as pd
3 import time
4
5 # importar a base de produtos
6 tabela = pd.read_csv("produtos.csv")
7 print(tabela)
8
```

```
      codigo  marca  ... custo  obs
0  MOL0000251  Logitech  ...   6.5  NaN
1  MOL0000192  Logitech  ...   5.0  NaN
2  CAHA000251   Hashtag  ...  11.0  NaN
3  CAHA000252   Hashtag  ...  11.0  Conferir estoque
4  MOMU000111  Multilaser  ...   3.4  NaN
..      ...      ...  ...   ...  ...
288 ACAP000192   Apple  ...   3.8  NaN
289 ACSA0009.3   Samsung  ...   2.1  NaN
290 CEM0000271   Motorola  ...  72.5  NaN
291 FOM0000152   Motorola  ...  33.0  NaN
292 CEM0000223   Motorola  ...  55.0  NaN
```

```
[293 rows x 7 columns]
```

Parte 3

# Biblioteca Pyautogui



# Biblioteca Pyautogui

## Instalação e Entendimento

Você deve ter visto que fizemos a importação da **biblioteca pyautogui**, essa é a biblioteca que vai permitir com que você controle o mouse e o seu teclado para fazer as automações no seu computador utilizando o Python.

Para fazer a instalação da biblioteca basta escrever **pip install pyautogui** no seu terminal.

Vou deixar aqui o link para a documentação dessa biblioteca, caso tenha dúvidas ou queira saber o que mais é possível fazer com essa biblioteca.

<https://pyautogui.readthedocs.io/en/latest/>

Nessa aula nós vamos utilizar basicamente 3 comandos da biblioteca Pyautogui:

- **Pyautogui.press** – Serve para pressionar uma tecla do seu teclado
- **Pyautogui.write** – Serve para escrever com o teclado (como se fosse você digitando)
- **Pyautogui.click** – Serve para clicar com o mouse

Em relação ao **pyautogui.click** você vai notar que precisamos passar uma **posição x e y** para que ele saiba onde tem que fazer o clique do mouse.

**IMPORTANTE:** Essa posição que nós vamos passar é em relação ao SEU monitor, porque ele leva em consideração o tamanho do monitor.

Dentro dos arquivos da aula nós temos um arquivo chamado **pegar\_posição.py**, que já tem um código para pegar a posição do seu mouse.

Com isso você vai ter a posição exata em relação ao seu computador para executar sua automação.

Ao executar o código, ele demora 5 segundos para pegar a posição para que você tenha tempo suficiente de colocar o mouse onde precisa clicar.

É muito importante que você pegue essas posições, pois se utilizar o padrão do gabarito é bem provável que dê errado e acabe clicando onde não deve.

# Obtendo a Posição do Mouse

```
1 import pyautogui
2 import time
3
4 time.sleep(5)
5 print(pyautogui.position()) # pegar posicao do mouse para saber onde clicar
6 pyautogui.scroll(-200) # testar quanto de scroll
```

Esse é o código do arquivo **pegar\_posição.py**, que é o código que te permite pegar a posição atual do seu mouse para que você saiba exatamente onde clicar na sua automação.

Esse passo é muito importante para que você consiga clicar no local correto, então sempre que for utilizar o **pyautogui** para clicar você precisa fazer esse procedimento.

É necessário, pois como a posição é em relação ao monitor, o que vamos utilizar pode ser diferente do seu, o que já seria um problema, pois o clique do mouse iria ocorrer em um local diferente.

Então sempre pegue essas informações antes de rodar o seu código para evitar de clicar em algo que não deva durante a execução do seu código.

Esse bloco de código não faz parte da nossa automação, ele é um código independente só para que você consiga pegar a posição do mouse nos momentos em que for clicar para garantir que vai clicar no local correto.

Parte 4

# Automação em Python com Pyautogui

# Automação em Python com Pyautogui

## Abrindo o Navegador

Agora que você já sabe o que vamos fazer e tem todos os arquivos e bibliotecas necessárias podemos dar início a nossa automação.

```
# define o tempo de espera entre os comandos do Pyautogui
pyautogui.PAUSE = 0.5

# abrir sistema (no nosso caso o chrome)
pyautogui.press("win")
pyautogui.write("chrome")
pyautogui.press("enter")
pyautogui.write("https://d1p.hashtagtreinamentos.com/python/intensivao/login")
pyautogui.press("enter")

# espera carregar
time.sleep(5)
```

Vamos começar utilizando o comando **pyautogui.PAUSE**, que é um comando para definir qual o tempo de espera entre os comandos do Pyautogui.

Isso quer dizer que entre cada comando dentro do seu código vamos ter esse tempo de espera. Isso é importante para que dê tempo de executar tudo de forma correta.

Em seguida nós temos o comando **pyautogui.press**, que é o comando para pressionar uma tecla do seu teclado.

Aqui temos a utilização do “win” que é a tecla de Windows do computador, para abirmos o menu iniciar.

Em seguida temos o comando **pyautogui.write**, que é o comando para escrever.

Então vamos escrever Chrome que é o navegador que vamos utilizar, mas você pode alterar para o seu navegador de preferência.

Vale lembrar que tudo aqui é como se você estivesse fazendo de forma manual, então pode ir fazendo o processo você mesmo para verificar quais são os passos utilizados.

Como é uma reprodução do que você faria pode fazer passo a passo para entender o que precisa fazer em cada etapa.

Essa parte inicial vamos abrir o navegador e escrever o link de onde vamos fazer o login para cadastrar os produtos.

Depois desse processo temos o **time.sleep** para esperar que o site carregue antes de começar a próxima etapa.

# Automação em Python com Pyautogui

## Fazendo o Login no Sistema

```
# fazer login (aqui pode preencher com qualquer dado de login)
pyautogui.click(x=1686, y=591)
pyautogui.write("pythonimpressionador@gmail.com")
pyautogui.press("tab")
pyautogui.write("sua senha")
pyautogui.click(x=1902, y=838)
```

Com o site/sistema aberto, nós podemos começar com o processo de login.

Nesse caso qualquer informação de login e senha vai funcionar, mas no sistema que for utilizar pessoal ou da empresa você precisará inserir as informações corretas.

**IMPORTANTE:** Aqui você vai notar que vamos utilizar o **pyautogui.click**, então é muito importante que já tenha as posições de onde vai clicar tanto para escrever o login quanto para clicar no botão para entrar no sistema.

Outro ponto que utilizamos aqui é o `pyautogui.press("tab")` que seria a utilização da tecla tab para passar para o próximo campo.

Isso é interessante, pois evita com que você tenha que utilizar o clique do mouse várias vezes, assim evita ter que ficar pegando a posição de cada campo.

Isso vai ser muito útil quando formos fazer o cadastro dos produtos, pois temos vários campos e você não vai querer pegar a posição de cada um deles, não é mesmo?

### Sistema Python PowerUp

#### Faça seu Login para acessar

**E-mail**

**Senha**

Logar

# Automação em Python com Pyautogui

## Cadastro dos Produtos

```
# aqui precisamos percorrer as linhas da tabela
# para cada linha vamos cadastrar um produto
for linha in tabela.index:
    pyautogui.click(x=1723, y=418) # clica no 1º campo
    pyautogui.write(str(tabela.loc[linha, "codigo"])) # pega o código da tabela e escreve no campo
    pyautogui.press("tab") # passa pro proximo campo
    # agora repete isso para os outros campos
    pyautogui.write(str(tabela.loc[linha, "marca"]))
    pyautogui.press("tab")
    pyautogui.write(str(tabela.loc[linha, "tipo"]))
    pyautogui.press("tab")
    pyautogui.write(str(tabela.loc[linha, "categoria"]))
    pyautogui.press("tab")
    pyautogui.write(str(tabela.loc[linha, "preco_unitario"]))
    pyautogui.press("tab")
    pyautogui.write(str(tabela.loc[linha, "custo"]))
    pyautogui.press("tab")
    if not pd.isna(tabela.loc[linha, "obs"]): # verifica se existe informação em obs, caso contrario não preenche
        pyautogui.write(str(tabela.loc[linha, "obs"]))
    #pyautogui.scroll(-200) # cuidado com o scroll,
    #ele não pode ir até o final da tela para o click da linha de baixo funcionar sempre
    pyautogui.click(x=1808, y=1380)
    pyautogui.scroll(5000)
```

Agora nós vamos fazer de fato o registro das informações no sistema.

Vamos iniciar com a estrutura de repetição for, que vai percorrer todas as linhas da base de dados para registrar cada uma das informações dos produtos.

Então vamos passar em cada linha da tabela e preencher todas as informações no sistema e clicar em enviar.

**OBS:** Temos o comando **pyautogui.scroll**, que vai servir para utilizar o scroll (bolinha) do mouse, mas pode ser que o comando antes de clicar não seja necessário. É bom fazer o teste manual para verificar o que acontece depois que começa os cadastros dos produtos.

# Automação em Python com Pyautogui

## Cadastro dos Produtos

```
for linha in tabela.index:
    pyautogui.click(x=1723, y=418) # clica no 1º campo
    pyautogui.write(str(tabela.loc[linha, "codigo"])) # pega o código da tabela e escreve no campo
    pyautogui.press("tab") # passa pro próximo campo
```

A estrutura de repetição for do jeito que está escrita, seria basicamente interpretada da seguinte maneira:

“Para cada linha/número no índice da tabela faça isso”

Onde o índice nada mais é do que o número da linha na tabela, então o código vai percorrer linha por linha e executar tudo o que estiver dentro dessa estrutura de repetição.

Temos o comando de click para clicar no primeiro campo onde vamos preencher a primeira informação e em seguida temos um comando um pouco diferente.

Pyautogui.write – Comando para escrever

Str – É um comando para transformar em string (texto)

.loc – É um comando para buscar uma informação, nesse caso dentro da nossa tabela. Então vamos buscar uma informação na linha x (dependendo de quantas vezes estamos repetindo) na coluna “codigo”.

Dessa forma vamos poder percorrer todas as informações da tabela e escrever no sistema a informação específica de cada campo.

Então no campo de **código do produto** vamos olhar na coluna código. No campo de **marca do produto** vamos buscar a informação na coluna marca e assim por diante.

A ideia é repetir o procedimento para todas as informações do produto para finalizar o cadastro.



## Automação em Python com Pyautogui

# Cadastro dos Produtos

```
if not pd.isna(tabela.loc[linha, "obs"]): # verifica se existe informação em obs, caso contrario não preenche
    pyautogui.write(str(tabela.loc[linha, "obs"]))
#pyautogui.scroll(-200) # cuidado com o scroll,
#ele não pode ir até o final da tela para o click da linha de baixo funcionar sempre
pyautogui.click(x=1808, y=1380)
pyautogui.scroll(5000)
```

Antes de utilizar o comando click para enviar e registrar as informações no sistema, vamos fazer uma breve verificação com a **estrutura condicional if** (estrutura se) para verificar se o campo de observação possui ou não alguma informação.

Vamos apenas buscar se a informação de observação não está vazia, se não estiver vamos preencher o campo de **obs**, caso contrário vamos seguir sem fazer nada. Isso é necessário, pois nem todos os produtos possuem essa informação.

Feito isso podemos clicar no botão de enviar para registrar as informações do produto. Depois disso a nossa estrutura de repetição vai continuar o mesmo processo até terminar todos os produtos da lista.

Como temos muitos produtos e temos aquele tempo entre os comandos do pyautogui, esse processo vai levar um tempo. Então caso queira fazer testes menores apenas para validação é interessante ajustar sua base de dados com 5 ou 10 informações só para garantir que tudo está certo.

Assim você consegue verificar se tem algum erro e ajustar antes de registrar os quase 300 produtos da lista!

# Parte 5

# Conclusão

# Conclusão

## Conclusão

A ideia desse projeto foi te mostrar como você pode **automatizar suas tarefas repetitivas utilizando o Python**. São de fato tarefas que você faz no dia a dia, seja em casa ou até mesmo no seu trabalho.

Já imaginou o tempo que isso vai economizar? Imagine você tendo que cadastrar quase que 300 produtos, com 6 ou 7 informações por produto.

Tomaria muito tempo, fora que você poderia errar na hora de escrever ou colar uma informação no campo errado. Com a automação você já faz essa verificação e com tudo certo vai repetir o processo corretamente para todas as informações.

Com isso você ganha muito tempo, pode focar em outras atividades e diminui drasticamente a chance de erros durante o processo de cadastro.

Fica o desafio pra você tentar reproduzir o que fizemos na aula e depois aplicar em um projeto seu ou até mesmo no seu trabalho para automatizar suas tarefas. Isso vai aumentar bastante sua produtividade, pois pode deixar a automação rodando enquanto faz outras atividades, enquanto sai para o almoço.

Já imaginou isso? Você sai pro almoço e deixa sua automação rodando, quando chegar já está tudo pronto enquanto as outras pessoas precisam fazer todo o processo de forma manual!

Espero que tenha gostado da aula e tenha visto como é útil e importante a automação de tarefas com **Python**. Fora que utilizando a biblioteca **pyautogui** fica muito mais fácil, pois vai simplesmente replicar os passos que você faria de forma manual!

# {JORNADA} PYTHON

100% GRATUITO E ONLINE

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



@hashtagprogramacao



youtube.com/hashtag-programacao

