**Python**

Criado em 1989, por Guido Van Rossum na Holanda. Inspirado na linguagem ABC, o python tambem possui parte da sintaxe derivada do C, compreensão de listas, funções anonimas e função map de Haskell, os iteradores são baseados na Icon, tratamentos de exceção e módulos da Modula-3, expressões regulares de Perl.

Guido era um grande fã do programa da BBC “Monty Python’s Flying Circus”, da onde se originou o nome de uma das linguagens mais utilizadas no mundo.

Recursos do Python:

### Uma linguagem interpretada

O Python é uma linguagem interpretada, o que significa que executa diretamente o código linha por linha. Se houver erros no código do programa, ele será interrompido. Portanto, os programadores podem encontrar rapidamente erros no código.

### Uma linguagem fácil de usar

O Python usa palavras semelhantes às do inglês. Ao contrário de outras linguagens de programação, o Python não usa chaves. Em vez disso, usa recuo.

### Uma linguagem com tipos dinâmicos

Os programadores não precisam declarar tipos de variáveis ao escrever o código, porque o Python os determina no tempo de execução. Dessa forma, é possível escrever programas Python mais rapidamente.

### Uma linguagem de alto nível

O Python está mais próximo das linguagens humanas do que outras linguagens de programação. Portanto, os programadores não precisam se preocupar com suas funcionalidades subjacentes, como arquitetura e gerenciamento de memória.

### Uma linguagem orientada a objetos

O Python considera tudo como um objeto, mas também aceita outros tipos de programação, como estruturada e funcional.

**Bibliotecas**

Por padrão, o Python vem com a Standard Library, que contém muitas funções reutilizáveis.Além disso, mais de 137 mil bibliotecas Python estão disponíveis para várias aplicações, incluindo desenvolvimento Web, ciência de dados e machine learning (ML). Uma biblioteca é uma coleção de códigos usados com frequência que os desenvolvedores podem incluir em seus programas Python para evitar escrever o código do zero.

As bibliotecas mais utilizadas

#### Matplotlib

Os desenvolvedores usam o Matplotlib para representar dados em gráficos bidimensionais e tridimensionais (2D e 3D) de alta qualidade. É frequentemente usado em aplicações científicas. Com o Matplotlib, é possível visualizar dados em diferentes gráficos, como de barras e de linhas. Você também pode criar vários gráficos de uma só vez, e é possível fazer a portabilidade dos gráficos entre todas as plataformas.

#### Pandas

O Pandas fornece estruturas de dados otimizadas e flexíveis para manipular dados de séries temporais e dados estruturados, como tabelas e matrizes. Por exemplo, use o Pandas para ler, gravar, mesclar, filtrar e agrupar dados. Muitas pessoas o usam para ciência de dados, análise de dados e tarefas de ML.

#### NumPy

O NumPy é uma biblioteca popular que os desenvolvedores usam para criar e gerenciar facilmente arrays, manipular formas lógicas e realizar operações de álgebra linear. É compatível com a integração com muitas linguagens, como C e C++.

#### Requests

A biblioteca Requests fornece funções úteis que são necessárias para o desenvolvimento Web. Você pode usá-la para enviar solicitações HTTP, adicionar cabeçalhos, adicionar parâmetros de URL, adicionar dados e executar muitas outras tarefas ao se comunicar com aplicações da Web.

#### OpenCV-Python

A OpenCV-Python é uma biblioteca que os desenvolvedores usam para processar imagens para aplicações de visão computacional. Ela fornece muitas funções para tarefas de processamento de imagens, como leitura e gravação de imagens simultaneamente, construção de um ambiente 3D com base em um ambiente 2D e captura e análise de imagens de vídeo.

#### Keras

A Keras é a biblioteca de rede neural profunda do Python, com excelente suporte para processamento de dados, visualização e muito mais. Ela é compatível com muitas redes neurais. Tem uma estrutura modular que oferece flexibilidade na escrita de aplicações de inovação.

**Importações**

Para importar uma biblioteca por inteiro utilize o comando import “nome da biblioteca”, para importar funções especificas da biblioteca utilize o comanndo from “nome da biblioteca” import “nome das funções”. Ao importamos uma biblioteca, a mesma pode ter um nome longo ou dificil de guardar, para facilitar utilize o comando import “nome da biblioteca” as “apelido da biblioteca”, assim você poderá utilizar-se de um ome mais facil e comum como nome da biblioteca.

**SQLite**

Python SQLite3 é um módulo integrado ao Python que permite a interação com o banco de dados SQLite3. O SQLite3 é um banco de dados leve, rápido e fácil de usar, que não requer instalação de um servidor separado. Ele é amplamente utilizado em aplicativos de desktop, mobile e web, devido à sua simplicidade e eficiência.

Sim, sqlite3faz parte da biblioteca padrão do Python para versões 2.5 e posteriores. Ele fornece uma interface SQL compatível com a especificação DB-API 2.0 descrita pelo PEP 249. Isso significa que você não precisa instalar nada extra se estiver usando essas versões do Python; você pode começar a usar bancos de dados SQLite em seus aplicativos imediatamente

### Facilidade de uso

### Portabilidade

### Baixo consumo de recursos

### Ampla adoção

* Autocontido
* Sem servidor (serverless)
* Configuração zero
* Transacional

**Código Fonte**

import tkinter as tk

from tkinter import ttk, messagebox

import sqlite3

#Eduardo Lucas Lemes Januário

#Sistema de Controle de Estoque

# FUNÇÕES Do BANCO DE DADOS

# Função para conectar ao banco de dados e criar a tabela

def criar\_banco():

conn = sqlite3.connect('estoque.db') # Criando o banco de dados

cursor = conn.cursor()

# Criando a tabela de produtos se ela não existir

cursor.execute('''

CREATE TABLE IF NOT EXISTS produtos (

codigo TEXT PRIMARY KEY,

nome TEXT NOT NULL,

quantidade INTEGER NOT NULL,

preco REAL NOT NULL

)

''')

conn.commit() # Confirmando a criação da tabela

conn.close() # Fechando a conexão

# Chamando a função para criar o banco de dados e a tabela

criar\_banco()

# Função para adicionar produto ao banco de dados

def salvar\_produto\_bd(*codigo*, *nome*, *quantidade*, *preco*):

conn = sqlite3.connect('estoque.db')

cursor = conn.cursor()

# Inserir o produto no banco de dados

try:

cursor.execute('''

INSERT INTO produtos (codigo, nome, quantidade, preco)

VALUES (?, ?, ?, ?)

''', (codigo, nome, quantidade, preco))

conn.commit() # Confirmando a inserção

messagebox.showinfo("Sucesso", f"Produto {nome} salvo com sucesso!")

except sqlite3.IntegrityError:

messagebox.showerror("Erro", f"Produto com código {codigo} já existe.")

conn.close() # Fechando a conexão

# Função para buscar produto no banco de dados

def buscar\_produto\_bd(*codigo*):

conn = sqlite3.connect('estoque.db')

cursor = conn.cursor()

cursor.execute("SELECT \* FROM produtos WHERE codigo = ?", (codigo,))

produto = cursor.fetchone()

conn.close() # Fechando a conexão

return produto

# Função para editar produto no banco de dados

def edit\_produto\_bd(*codigo*, *nome*, *quantidade*, *preco*):

conn = sqlite3.connect('estoque.db')

cursor = conn.cursor()

try:

cursor.execute("UPDATE produtos SET nome = ?, quantidade = ?, preco = ? WHERE codigo = ?",

(nome, quantidade, preco, codigo))

conn.commit() # Confirmando a atualização

messagebox.showinfo("Sucesso", "Produto editado com sucesso!")

except sqlite3.Error:

messagebox.showerror("Erro", "Falha ao editar produto!")

conn.close() # Fechando a conexão

# Função para excluir produto no banco de dados

def delete\_produto\_bd(*codigo*):

conn = sqlite3.connect('estoque.db')

cursor = conn.cursor()

try:

cursor.execute("DELETE FROM produtos WHERE codigo = ?", (codigo,))

conn.commit() # Confirmando a exclusão

if cursor.rowcount > 0:

messagebox.showinfo("Sucesso", "Produto excluído com sucesso!")

else:

messagebox.showwarning("Aviso", "Produto não encontrado!")

except sqlite3.Error:

messagebox.showerror("Erro", "Falha ao excluir produto!")

conn.close() # Fechando a conexão

# FUNÇÕES DA INTERFACE

#Função para selecionar o menu com base no acesso

def configurar\_abas():

nivel\_acesso = cod\_acess\_entry.get()

# Exibe apenas as abas permitidas

if nivel\_acesso == '01': # Usuário com acesso limitado

notebook.add(aba\_buscar, *text*="Menu Usuário")

notebook.forget(aba\_acesso)

elif nivel\_acesso == '02': # Administrador com acesso total

notebook.add(aba\_menu, *text*="Menu Admin")

notebook.forget(aba\_acesso)

#Função botão add da busca

def chama\_add():

notebook.add(aba\_adicionar, *text*="Adicionar Produto")

notebook.forget(aba\_buscar)

#Função botão busca do menu-admin

def chama\_buscar\_admin():

notebook.add(aba\_buscar, *text*="Buscar Produto")

notebook.forget(aba\_menu)

#Função botão edit do menu-admin

def chama\_edit\_admin():

notebook.add(aba\_edit, *text*="Editar Produto")

notebook.forget(aba\_menu)

#Função botão fechar da aba add

def exit\_add():

notebook.forget(aba\_adicionar)

notebook.add(aba\_acesso, *text*="Acesso")

#Função botão fechar da aba busca

def exit\_busca():

notebook.forget(aba\_buscar)

notebook.add(aba\_acesso, *text*="Acesso")

#Função botão fechar da menu

def exit\_menu():

notebook.forget(aba\_menu)

notebook.add(aba\_acesso, *text*="Acesso")

#Função botão fechar da aba edit

def exit\_edit():

notebook.forget(aba\_edit)

notebook.add(aba\_acesso, *text*="Acesso")

# Função para obter dados da interface e adicionar produto ao estoque

def adicionar\_produto\_interface():

codigo = cod\_entry.get()

nome = name\_entry.get()

try:

quantidade = *int*(qtd\_entry.get())

preco = *float*(preco\_entry.get())

except *ValueError*:

messagebox.showerror("Erro", "Quantidade e preço devem ser números válidos.")

return

# Jogando dados pra função add BD

salvar\_produto\_bd(codigo, nome, quantidade, preco)

# Limpa os campos da interface

cod\_entry.delete(0, tk.END)

name\_entry.delete(0, tk.END)

qtd\_entry.delete(0, tk.END)

preco\_entry.delete(0, tk.END)

# Função para buscar e exibir produto da aba busca

def busca\_produto\_interface\_busca():

codigo = cod\_busca\_entry.get()

produto = buscar\_produto\_bd(codigo)

if produto:

# Exibir os dados do produto na interface de busca

nome\_busca\_entry.config(*state*='normal') # Permite edição

nome\_busca\_entry.delete(0, tk.END)

nome\_busca\_entry.insert(0, produto[1]) # Nome

nome\_busca\_entry.config(*state*='readonly') # Modo somente leitura

qtd\_busca\_entry.config(*state*='normal')

qtd\_busca\_entry.delete(0, tk.END)

qtd\_busca\_entry.insert(0, produto[2]) # Quantidade

qtd\_busca\_entry.config(*state*='readonly')

preco\_busca\_entry.config(*state*='normal')

preco\_busca\_entry.delete(0, tk.END)

preco\_busca\_entry.insert(0, produto[3]) # Preço

preco\_busca\_entry.config(*state*='readonly')

else:

messagebox.showerror("Erro", f"Produto com código {codigo} não encontrado.")

#Função para buscar e exibir produto da aba edit

def busca\_produto\_interface\_edit():

codigo = cod\_edit\_entry.get()

produto = buscar\_produto\_bd(codigo)

if produto:

# Exibir os dados do produto na interface de busca

nome\_edit\_entry.config(*state*='normal') # Permite edição

nome\_edit\_entry.delete(0, tk.END)

nome\_edit\_entry.insert(0, produto[1]) # Nome

nome\_edit\_entry.config(*state*='readonly') # Modo somente leitura

qtd\_edit\_entry.config(*state*='normal')

qtd\_edit\_entry.delete(0, tk.END)

qtd\_edit\_entry.insert(0, produto[2]) # Quantidade

qtd\_edit\_entry.config(*state*='readonly')

preco\_edit\_entry.config(*state*='normal')

preco\_edit\_entry.delete(0, tk.END)

preco\_edit\_entry.insert(0, produto[3]) # Preço

preco\_edit\_entry.config(*state*='readonly')

else:

messagebox.showerror("Erro", f"Produto com código {codigo} não encontrado.")

nome\_edit\_entry.config(*state*='normal') # Permite edição

qtd\_edit\_entry.config(*state*='normal')

preco\_edit\_entry.config(*state*='normal')

# Função para buscar e atualizar o produto

def editar\_produto\_interface():

codigo = cod\_edit\_entry.get()

nome = nome\_edit\_entry.get()

try:

quantidade = *int*(qtd\_edit\_entry.get())

preco = *float*(preco\_edit\_entry.get())

except *ValueError*:

messagebox.showerror("Erro", "Quantidade e preço devem ser números válidos.")

return

# Jogando dados pra função edit BD

edit\_produto\_bd(codigo, nome, quantidade, preco)

# Limpa os campos da interface

cod\_busca\_entry.delete(0, tk.END)

nome\_busca\_entry.config(*state*='normal')

nome\_busca\_entry.delete(0, tk.END)

qtd\_busca\_entry.config(*state*='normal')

qtd\_busca\_entry.delete(0, tk.END)

preco\_busca\_entry.config(*state*='normal')

preco\_busca\_entry.delete(0, tk.END)

# Função para buscar e excluir o produto

def deletar\_produto\_interface():

codigo = cod\_edit\_entry.get()

# Jogando dados pra função delete BD

delete\_produto\_bd(codigo)

# Limpa os campos da interface

cod\_edit\_entry.delete(0, tk.END)

nome\_edit\_entry.config(*state*='normal')

nome\_edit\_entry.delete(0, tk.END)

qtd\_edit\_entry.config(*state*='normal')

qtd\_edit\_entry.delete(0, tk.END)

preco\_edit\_entry.config(*state*='normal')

preco\_edit\_entry.delete(0, tk.END)

# INTERFACE

# Configuração da interface

tela = tk.Tk()

tela.title("Controle de Estoque")

# Criando o notebook (abas)

notebook = ttk.Notebook(tela)

notebook.pack(*pady*=10, *expand*=True)

# Aba de Acesso

aba\_acesso = ttk.Frame(notebook)

notebook.add(aba\_acesso, *text*="Acesso")

# Elementos da aba de acesso

title\_label = tk.Label(aba\_acesso, *text*="Acesso ao Sistema ")

title\_label.grid(*row*=0, *column*=1, *padx*=10, *pady*=10)

user\_label = tk.Label(aba\_acesso, *text*="01 - usuário ")

user\_label.grid(*row*=1, *column*=0, *padx*=10, *pady*=10)

adm\_label = tk.Label(aba\_acesso, *text*="02 - ADM ")

adm\_label.grid(*row*=1, *column*=2, *padx*=10, *pady*=10)

cod\_acess\_label = tk.Label(aba\_acesso, *text*="Código do Acesso: ")

cod\_acess\_label.grid(*row*=2, *column*=0, *padx*=10, *pady*=6)

cod\_acess\_entry = tk.Entry(aba\_acesso)

cod\_acess\_entry.grid(*row*=3, *column*=1, *padx*=10, *pady*=6)

# Botão para acesso edit produto

botao\_acess = tk.Button(aba\_acesso, *text*="Acessar", *command*=configurar\_abas)

botao\_acess.grid(*row*=4, *column*=2, *columnspan*=1, *pady*=10)

# Aba de Menu - Admin

aba\_menu = ttk.Frame(notebook)

notebook.add(aba\_menu, *text*="Menu - Admin")

# Elementos da aba de adicionar

botao\_busca\_menu\_admin = tk.Button(aba\_menu, *text*="Buscar Produto", *command*=chama\_buscar\_admin)

botao\_busca\_menu\_admin.grid(*row*=0, *column*=0, *columnspan*=2, *pady*=10)

botao\_edit\_menu\_admin = tk.Button(aba\_menu, *text*="Editar Produto", *command*=chama\_edit\_admin)

botao\_edit\_menu\_admin.grid(*row*=0, *column*=4, *columnspan*=2, *pady*=10)

botao\_fechar\_busca = tk.Button(aba\_menu, *text*="fechar", *command*=exit\_menu)

botao\_fechar\_busca.grid(*row*=1, *column*=2, *columnspan*=2, *pady*=10)

# Aba de Buscar Produto

aba\_buscar = ttk.Frame(notebook)

notebook.add(aba\_buscar, *text*="Buscar Produto")

# Elementos da aba de buscar

# Botão para buscar produto

botao\_fechar\_busca = tk.Button(aba\_buscar, *text*="X", *command*=exit\_busca)

botao\_fechar\_busca.grid(*row*=0, *column*=3, *columnspan*=2, *pady*=10)

cod\_busca\_label = tk.Label(aba\_buscar, *text*="Código do Produto: ")

cod\_busca\_label.grid(*row*=0, *column*=0, *padx*=10, *pady*=6)

cod\_busca\_entry = tk.Entry(aba\_buscar)

cod\_busca\_entry.grid(*row*=0, *column*=1, *padx*=10, *pady*=6)

# Botão para buscar produto

botao\_buscar = tk.Button(aba\_buscar, *text*="Buscar Produto", *command*=busca\_produto\_interface\_busca)

botao\_buscar.grid(*row*=1, *column*=1, *columnspan*=2, *pady*=10)

nome\_busca\_label = tk.Label(aba\_buscar, *text*="Nome do Produto: ")

nome\_busca\_label.grid(*row*=2, *column*=0, *padx*=10, *pady*=6)

nome\_busca\_entry = tk.Entry(aba\_buscar, *state*='readonly')

nome\_busca\_entry.grid(*row*=2, *column*=1, *padx*=10, *pady*=6)

qtd\_busca\_label = tk.Label(aba\_buscar, *text*="Quantidade do Produto: ")

qtd\_busca\_label.grid(*row*=3, *column*=0, *padx*=10, *pady*=6)

qtd\_busca\_entry = tk.Entry(aba\_buscar, *state*='readonly')

qtd\_busca\_entry.grid(*row*=3, *column*=1, *padx*=10, *pady*=6)

preco\_busca\_label = tk.Label(aba\_buscar, *text*="Preço do Produto: ")

preco\_busca\_label.grid(*row*=4, *column*=0, *padx*=10, *pady*=6)

preco\_busca\_entry = tk.Entry(aba\_buscar, *state*='readonly')

preco\_busca\_entry.grid(*row*=4, *column*=1, *padx*=10, *pady*=6)

# Botão para adicionar produto

botao\_add\_busca = tk.Button(aba\_buscar, *text*="Adicionar Produto", *command*=chama\_add)

botao\_add\_busca.grid(*row*=5, *column*=1, *columnspan*=2, *pady*=10)

# Aba de Adicionar Produto

aba\_adicionar = ttk.Frame(notebook)

notebook.add(aba\_adicionar, *text*="Adicionar Produto")

# Elementos da aba de adicionar

botao\_fechar\_add = tk.Button(aba\_adicionar, *text*="X", *command*=exit\_add)

botao\_fechar\_add.grid(*row*=0, *column*=3, *columnspan*=2, *pady*=10)

cod\_label = tk.Label(aba\_adicionar, *text*="Código do Produto: ")

cod\_label.grid(*row*=0, *column*=0, *padx*=10, *pady*=6)

cod\_entry = tk.Entry(aba\_adicionar)

cod\_entry.grid(*row*=0, *column*=1, *padx*=10, *pady*=6)

name\_label = tk.Label(aba\_adicionar, *text*="Nome do Produto: ")

name\_label.grid(*row*=1, *column*=0, *padx*=10, *pady*=6)

name\_entry = tk.Entry(aba\_adicionar)

name\_entry.grid(*row*=1, *column*=1, *padx*=10, *pady*=6)

qtd\_label = tk.Label(aba\_adicionar, *text*="Quantidade do Produto: ")

qtd\_label.grid(*row*=2, *column*=0, *padx*=10, *pady*=6)

qtd\_entry = tk.Entry(aba\_adicionar)

qtd\_entry.grid(*row*=2, *column*=1, *padx*=10, *pady*=6)

preco\_label = tk.Label(aba\_adicionar, *text*="Preço do Produto: ")

preco\_label.grid(*row*=3, *column*=0, *padx*=10, *pady*=6)

preco\_entry = tk.Entry(aba\_adicionar)

preco\_entry.grid(*row*=3, *column*=1, *padx*=10, *pady*=6)

# Botão para adicionar produto

botao\_adicionar = tk.Button(aba\_adicionar, *text*="Adicionar Produto", *command*=adicionar\_produto\_interface)

botao\_adicionar.grid(*row*=4, *column*=0, *columnspan*=2, *pady*=10)

# Aba de Editar Produto

aba\_edit = ttk.Frame(notebook)

notebook.add(aba\_edit, *text*="Editar Produto")

# Elementos da aba de editar

botao\_fecha\_edit = tk.Button(aba\_edit, *text*="X", *command*=exit\_edit)

botao\_fecha\_edit.grid(*row*=0, *column*=3, *columnspan*=2, *pady*=10)

cod\_edit\_label = tk.Label(aba\_edit, *text*="Código do Produto: ")

cod\_edit\_label.grid(*row*=0, *column*=0, *padx*=10, *pady*=6)

cod\_edit\_entry = tk.Entry(aba\_edit)

cod\_edit\_entry.grid(*row*=0, *column*=1, *padx*=10, *pady*=6)

# Botão para editar produto

botao\_buscar\_edit = tk.Button(aba\_edit, *text*="Buscar Produto", *command*=busca\_produto\_interface\_edit)

botao\_buscar\_edit.grid(*row*=1, *column*=1, *columnspan*=2, *pady*=10)

nome\_edit\_label = tk.Label(aba\_edit, *text*="Nome do Produto: ")

nome\_edit\_label.grid(*row*=2, *column*=0, *padx*=10, *pady*=6)

nome\_edit\_entry = tk.Entry(aba\_edit, *state*='normal')

nome\_edit\_entry.grid(*row*=2, *column*=1, *padx*=10, *pady*=6)

qtd\_edit\_label = tk.Label(aba\_edit, *text*="Quantidade do Produto: ")

qtd\_edit\_label.grid(*row*=3, *column*=0, *padx*=10, *pady*=6)

qtd\_edit\_entry = tk.Entry(aba\_edit, *state*='normal')

qtd\_edit\_entry.grid(*row*=3, *column*=1, *padx*=10, *pady*=6)

preco\_edit\_label = tk.Label(aba\_edit, *text*="Preço do Produto: ")

preco\_edit\_label.grid(*row*=4, *column*=0, *padx*=10, *pady*=6)

preco\_edit\_entry = tk.Entry(aba\_edit, *state*='normal')

preco\_edit\_entry.grid(*row*=4, *column*=1, *padx*=10, *pady*=6)

# Botão para editar produto

botao\_edit\_edit = tk.Button(aba\_edit, *text*="Editar Produto", *command*=editar\_produto\_interface)

botao\_edit\_edit.grid(*row*=5, *column*=0, *pady*=10)

# Botão para excluir produto

botao\_delete = tk.Button(aba\_edit, *text*="Excluir Produto", *command*=deletar\_produto\_interface)

botao\_delete.grid(*row*=5, *column*=3, *pady*=10)

# Oculta todas as abas primeiro

notebook.forget(aba\_edit)

notebook.forget(aba\_adicionar)

notebook.forget(aba\_buscar)

notebook.forget(aba\_menu)

# Mantém a janela aberta

tela.mainloop()

**Conclusão**

O desenvolvimento do sistema de controle de estoque enfrentou desafios técnicos, como a integração de bibliotecas e a implementação de uma interface eficiente para diferentes níveis de usuários. Problemas como a navegação entre abas e a manipulação de erros foram superados com ajustes constantes. No futuro, o sistema pode ser aprimorado com funcionalidades como relatórios automatizados, integração com plataformas online e melhorias na interface e banco de dados. Essas melhorias garantiriam maior escalabilidade e usabilidade, mantendo o sistema relevante e eficaz.

**Bibliografia**

24/10/2024 as 10:19

<https://aws.amazon.com/pt/what-is/python/>

<https://pt.wikipedia.org/wiki/Python>

<https://awari.com.br/python-sqlite3-aprenda-a-utilizar-o-banco-de-dados-sqlite3-com-python/>

<https://www.geeksforgeeks.org/python-sqlite/>