Centro Universitário de Excelência
Sistemas de Informação

FoodDelivery com Listas e Filas

Autor(es): Dênis Maciel;
Douglas Pereira;
Graciano Marôto;
José Henrique Dias Moreira;
Lara Nunes Silva.

Vitória da Conquista/BA, Outubro - 2025

Objetivo

Apresentar o desenvolvimento de um sistema de gerenciamento de pedidos e produtos em Python, utilizando listas e filas como principais estruturas de dados para organizar e controlar informações de forma eficiente.

Resumo

Este trabalho apresenta o desenvolvimento de um sistema de gerenciamento de pedidos e produtos em Python, utilizando listas e filas como estruturas de dados principais. O sistema permite cadastrar itens, atualizar informações, controlar estoque e gerenciar pedidos, desde a criação até a finalização. O foco foi demonstrar como essas estruturas organizam informações de forma eficiente e prática.

Introdução

Estruturas de dados são fundamentais na programação, pois determinam a forma como informações são organizadas e acessadas. No contexto de sistemas de gerenciamento, listas e filas desempenham papel essencial: listas permitem armazenar, buscar e atualizar itens de forma flexível, enquanto filas garantem que pedidos sejam atendidos em ordem de chegada. O objetivo foi aplicar esses conceitos no desenvolvimento de um sistema de pedidos que simula um cenário real de delivery.

Tópicos

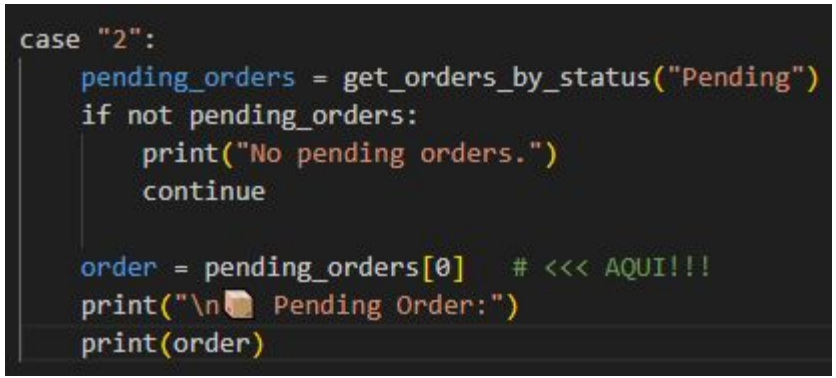
- O que é a **Fila** e como está sendo aplicada ao código;
- O que é **Lista** e como está sendo aplicada ao código;
- Fluxo do sistema
- Funcionamento do sistema
- Considerações finais
- Bibliografia

Fila -Teoria e prática aplicada

O que é a Fila?

Fila é uma estrutura de dados que segue o princípio FIFO (First In, First Out), onde o primeiro elemento a entrar é o primeiro a sair, sem acesso direto a elementos do meio.

- Exemplo (resumido) aplicado em código:



```
case "2":
    pending_orders = get_orders_by_status("Pending")
    if not pending_orders:
        print("No pending orders.")
        continue

    order = pending_orders[0]   # <<< AQUI!!!
    print("\n📦 Pending Order:")
    print(order)
```

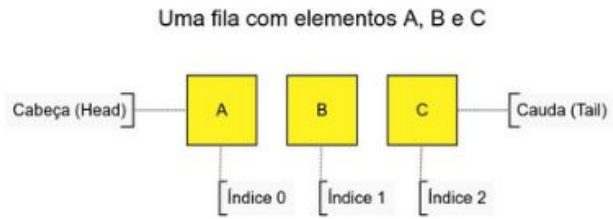
Fila -Teoria e prática aplicada

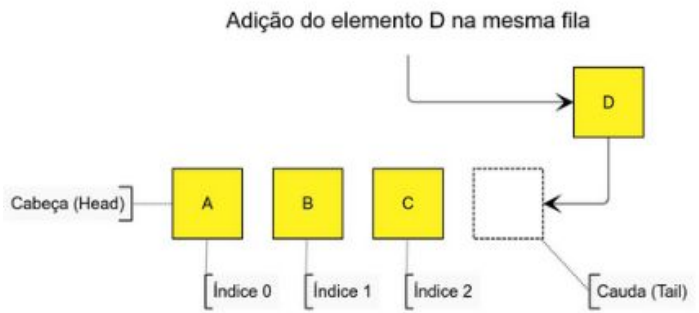
Como este trecho de código funciona?

- O sistema pega todos os pedidos com status "Pending" → isso é a fila de espera.
- Depois, acessa sempre o primeiro da lista (pending_orders[0]).

Este é o comportamento clássico de fila (FIFO – First In, First Out):

- O primeiro pedido que entrou como pendente é o primeiro a ser tratado.





Fila -Teoria e prática aplicada

Regras da Fila

- **Ordem de chegada importa** → os pedidos ficam armazenados em all_orders na ordem em que foram feitos.
- **Filtragem** → pending_orders = get_orders_by_status("Pending") cria a lista com os pedidos pendentes.
- **Atendimento sequencial** → order = pending_orders[0] garante que sempre o primeiro pedido pendente será tratado antes dos demais.
- **FIFO (First In, First Out)** → o primeiro pedido que entrou pendente é o primeiro a sair (ser aceito ou rejeitado).
- **Processamento contínuo** → a fila só anda quando o primeiro da lista é atendido; os outros aguardam sua vez.

Lista - Teoria e prática aplicada

O que é Lista?

Lista é uma estrutura de dados ordenada que permite acessar, inserir e remover elementos em qualquer posição usando índices.

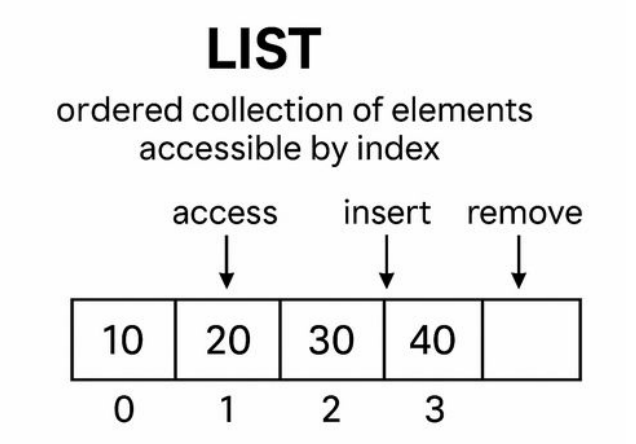
- Exemplo (resumido) aplicado em código:

```
all_orders = []
order_index = int(input("Select an order by code: ".center(width))) - 1
order = all_orders[order_index]
```

Lista - Teoria e prática aplicada

Como este trecho de código funciona?

- Armazenam todos os pedidos em all_orders.
- Permitem acesso por índice: exemplo, all_orders[0] pega o primeiro pedido.
- Solicitam ao usuário a escolha de um pedido, ao digitar o número/código, converte essa entrada em número inteiro.
- Selecionam o pedido correspondente order = all_orders[order_index]
- Pega da lista o pedido que corresponde à escolha do usuário.



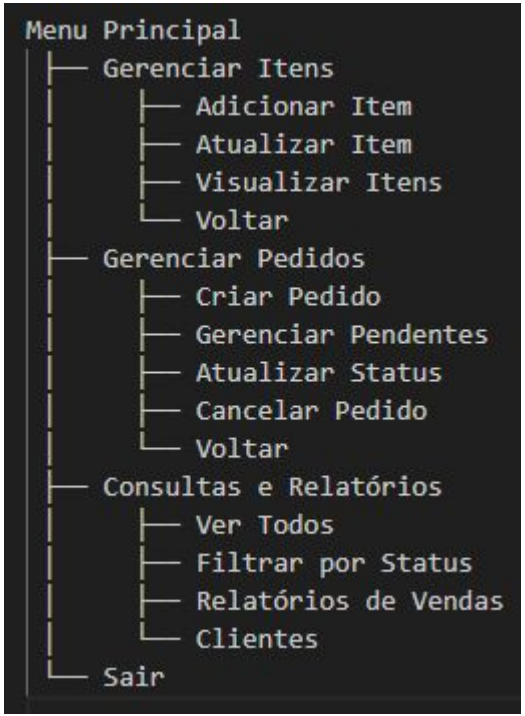
Lista -Teoria e prática aplicada

Regras da Lista

- **Estrutura ordenada:** os elementos ficam na sequência em que são inseridos.
- **Acesso direto por índice:** você pode pegar qualquer elemento pela posição (lista[0], lista[1], etc.).
- **Aceita duplicados:** pode ter pedidos iguais sem problema.
- **Inserção e remoção flexíveis:** você pode adicionar no fim (append), em qualquer posição (insert), ou remover (remove, pop).
- **Dinâmica:** aumenta e diminui conforme adiciona ou remove pedidos.

Listagem de fluxo do sistema

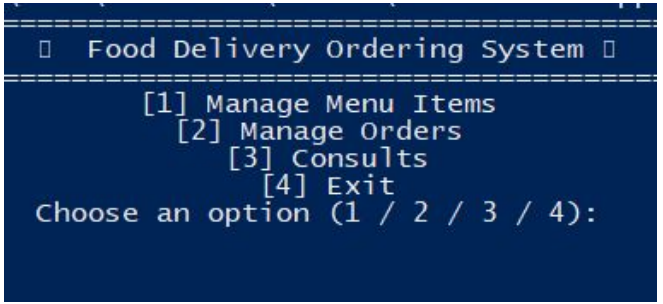
O diagrama em árvore apresentado no slide representa de forma clara e objetiva como o sistema está estruturado. A utilização de **listas** permite o gerenciamento eficiente de dados como itens, clientes e pedidos, enquanto as **filas** garantem que os pedidos sejam processados em ordem, simulando a operação real de um sistema de entrega de alimentos.



Funcionamento do sistema

Menu principal:

O menu principal permite gerenciar itens do cardápio, pedidos de clientes e gerar relatórios. A estrutura usa **listas** para armazenar dados e simulação de **filas** para controlar o fluxo dos pedidos, seguindo etapas como “Aceito”, “Preparando” e “Entregue”. As imagens mostram tanto o fluxo resumido do sistema (código), quanto a interface simples no terminal.

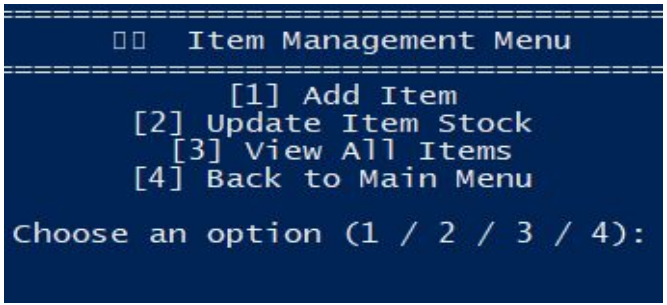



```
def main_menu():
    while True:
        print("\n" + "=" * width)
        print("Main Menu".center(width))
        print("=" * width)
        print("1. Manage Menu Items")
        print("2. Manage Orders")
        print("3. Consults")
        print("4. Exit")
        print("=" * width)
        choice = input("Select an option: ".center(width))
```

Funcionamento do sistema

Gerenciamento de itens:

O sistema usa listas para armazenar e gerenciar os itens do cardápio, permitindo adicionar, editar e visualizar produtos. Para os pedidos, utiliza simulação filas que garantem a ordem de atendimento e possibilitam o acompanhamento do status de cada pedido.

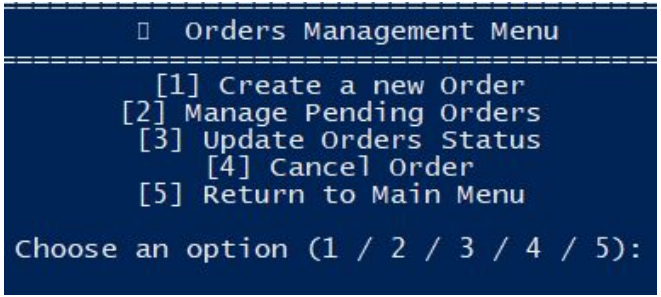



```
def manage_menu_items():
    while True:
        print("\n" + "=" * width)
        print("Manage Menu Items".center(width))
        print("=" * width)
        print("1. Add Item to Menu")
        print("2. Update Item")
        print("3. View All Items")
        print("4. Back to Main Menu")
        print("=" * width)
        choice = input("Select an option: ".center(width))
```

Funcionamento do sistema

Gerenciamento de pedidos:

O gerenciamento de pedidos é a funcionalidade do sistema dedicada a controlar todo o ciclo de um pedido, desde sua criação até a finalização. Nele, o usuário pode registrar novos pedidos, vinculando-os a um cliente e aos itens escolhidos; gerenciar pedidos pendentes, aceitando ou rejeitando conforme a disponibilidade; atualizar o status de cada pedido, acompanhando as etapas como preparo, entrega e conclusão; e cancelar pedidos ainda não finalizados, quando necessário.

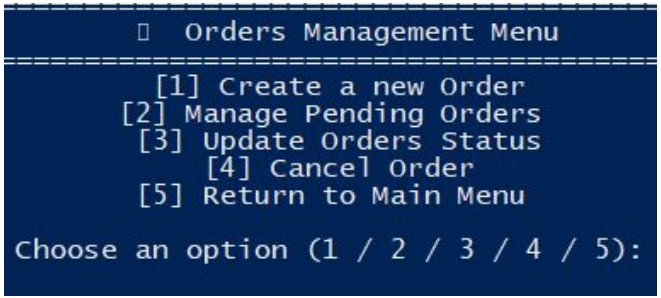



```python
def manage_orders():
    while True:
        print("\n" + "=" * width)
        print("Manage Orders".center(width))
        print("=" * width)
        print("1. Create New Order")
        print("2. Manage Pending Orders")
        print("3. Update Order Status")
        print("4. Cancel Order")
        print("5. Back to Main Menu")
        print("=" * width)
        choice = input("Select an option: ".center(width))
```

Funcionamento do sistema

Relatórios:

No sistema, a funcionalidade de relatórios está disponível no menu de consultas, na opção Sales Report. Ela permite acompanhar o desempenho das vendas de forma simples. A opção All registers exibe a quantidade total de pedidos já realizados, somando seus valores independentemente do status em que se encontram. Já a opção Closed sales considera apenas os pedidos concluídos e entregues, mostrando a quantidade e o valor arrecadado exclusivamente dessas vendas finalizadas.




```
def manage_orders():
    while True:
        print("\n" + "=" * width)
        print("Manage Orders".center(width))
        print("=" * width)
        print("1. Create New Order")
        print("2. Manage Pending Orders")
        print("3. Update Order Status")
        print("4. Cancel Order")
        print("5. Back to Main Menu")
        print("=" * width)
        choice = input("Select an option: ".center(width))
```

Considerações finais

O projeto trouxe desafios valiosos, como criar uma interface de terminal interativa e lidar com situações como entradas incorretas ou falta de estoque. O uso de classes deixou o código organizado e próximo de uma aplicação real. As listas foram essenciais para armazenar e manipular itens, clientes e pedidos, enquanto as filas refletiram a ordem natural de atendimento, garantindo que os pedidos fossem processados de forma justa e organizada.

Referências

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. *Algoritmos: teoria e prática*. 3. ed. Rio de Janeiro: LTC, 2014.

GOODRICH, Michael T.; TAMASSIA, Roberto. *Estruturas de dados e algoritmos em Python*. São Paulo: Bookman, 2016.

PYTHON SOFTWARE FOUNDATION. *Documentação Python*. Disponível em: https://docs.python.org/3/