Aula 07

Filas

Algoritmos e Estrutura de Dados II

2° Semestre - CDN

Prof. Dr. Dilermando Piva Jr.

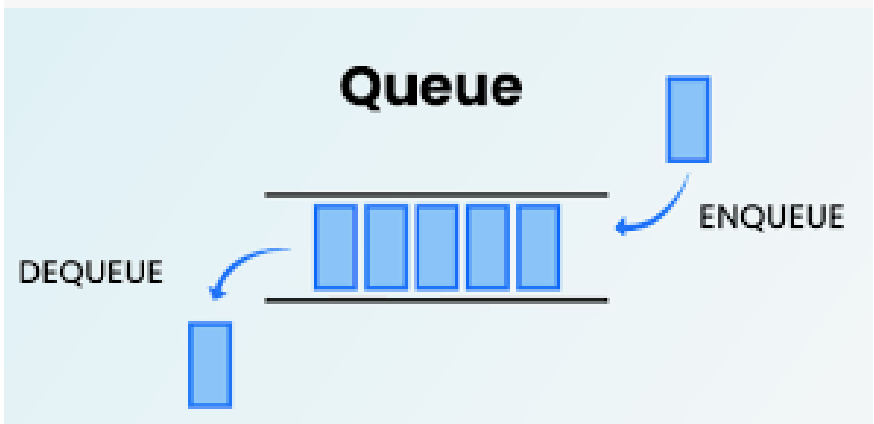
Conteúdo Programático - Planejamento

Semana	Data	Temas/Atividades
1	07/08	Acolhimento e Boas-vindas! Introdução a Disciplina. Formas de Avaliação e Percurso Pedagógico.
2	14/08	Tipo de dado abstrato. Introdução a Estrutura de Dados.
3	21/08	Complexidade de Algoritmos
4	28/08	Vetores não-Ordenados e busca sequencial
5	04/09	Vetores Ordenados e busca binária
6	11/09	Revisão de Programação Orientada a Objetos (POO)
7	18/09	Pilhas
8	25/09	Filas
9	02/10	Listas encadeadas
10	09/10	Recursão
11	*16/10*	***Primeira Avaliação Formal. (P1).*** *Correção da Avaliação após o intervalo.*
12	18/10	Algoritmos de Ordenação
13	23/10	Algoritmos de Ordenação
14	30/10	Árvores
15	06/11	Grafos
16	13/11	***Segunda Avaliação Formal (P2).*** Correção da Avaliação após o intervalo
17	27/11	***Apresentação PI do curso de CDN***
18	*04/12*	Tabela Hash (tabela de espalhamento) – Tópico extra.
19	*11/12*	***Exame / Avaliação Substitutiva.*** Correção da Avaliação após o intervalo. Finalização Disciplina
20	18/12	Finalização da disciplina.

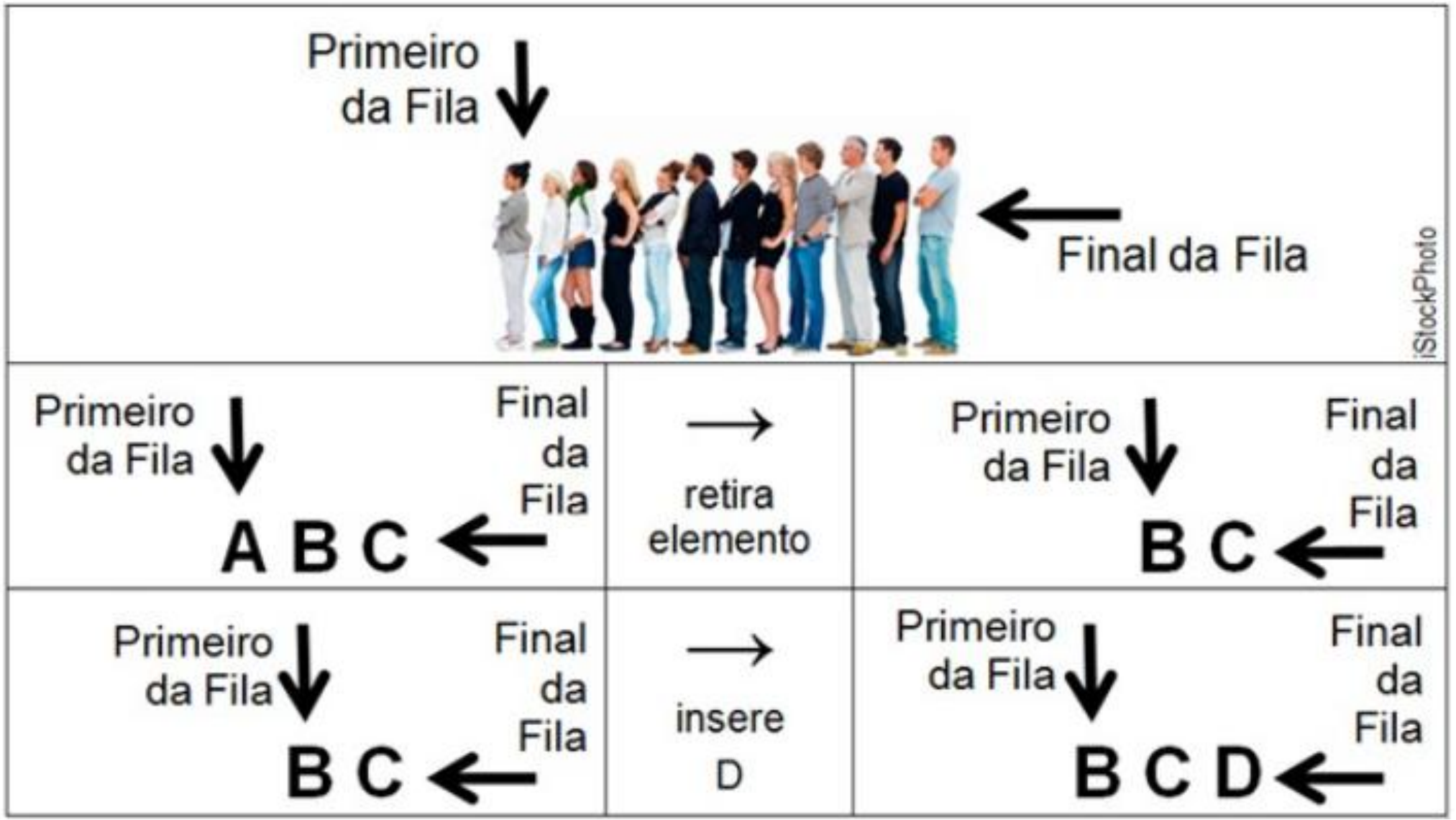
Você já passou por essa situação?

FILAS

- Uma fila é uma estrutura de dados linear em que a inserção de elementos é realizada em uma extremidade (final) e a remoção é realizada na outra extremidade (início).
- Assim como em uma fila de pessoas no mundo real, o primeiro a entrar será o primeiro a sair, o que define o princípio de ordenação FIFO, ou do inglês, First In First Out.
- Além disso, filas devem ser restritivas no sentido de que um elemento não pode passar na frente de seu antecessor.



FILAS



FILAS

```
x = Q.dequeue()
```


```
Q.enqueue(21)
```


```
Q.enqueue(74)
```

Atividade com IA

- Para se aprofundar mais...:
  - No conceito de FILAS...

- Faça individualmente, e depois compartilhe com o seu colega esses conceitos.

Peça para a IA:

Contexto: *Sou estudante de ciência de dados aprendendo estrutura de dados. Estou focado no conceito de* ***Filas (Queue)*** *e quero uma explicação abrangente que inclua:* ***Definição formal:*** *O que é uma fila? Quais são suas características principais (FIFO, operações básicas, etc.)?* ***Analogias do mundo real:*** *Exemplos concretos de como as filas funcionam em situações cotidianas.* ***Estrutura digital:*** *Como as filas são implementadas em programação? Explique com exemplos de código (em Python, se possível) as operações básicas.* ***Aplicações em ciência de dados e computação:*** *Onde as filas são usadas em algoritmos, processamento de dados ou em bibliotecas/frameworks de data science?* ***Vantagens e desvantagens:*** *Quando usar uma fila? Quais são as limitações?* ***Comparação com outras estruturas:*** *Como a fila se diferencia de uma pilha ou de uma lista encadeada?*

Por favor, explique de forma clara, com exemplos práticos e detalhes que ajudem a fixar o conceito.

FILAS

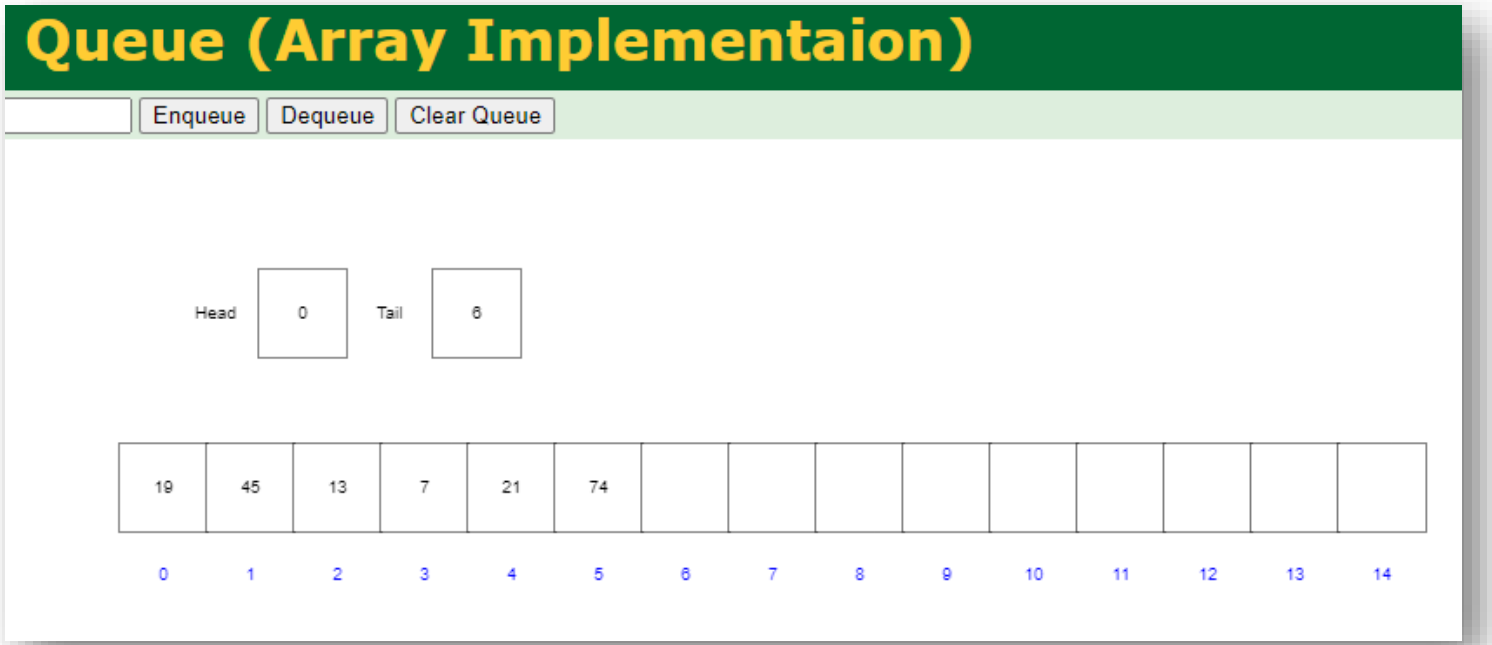
Operações básicas

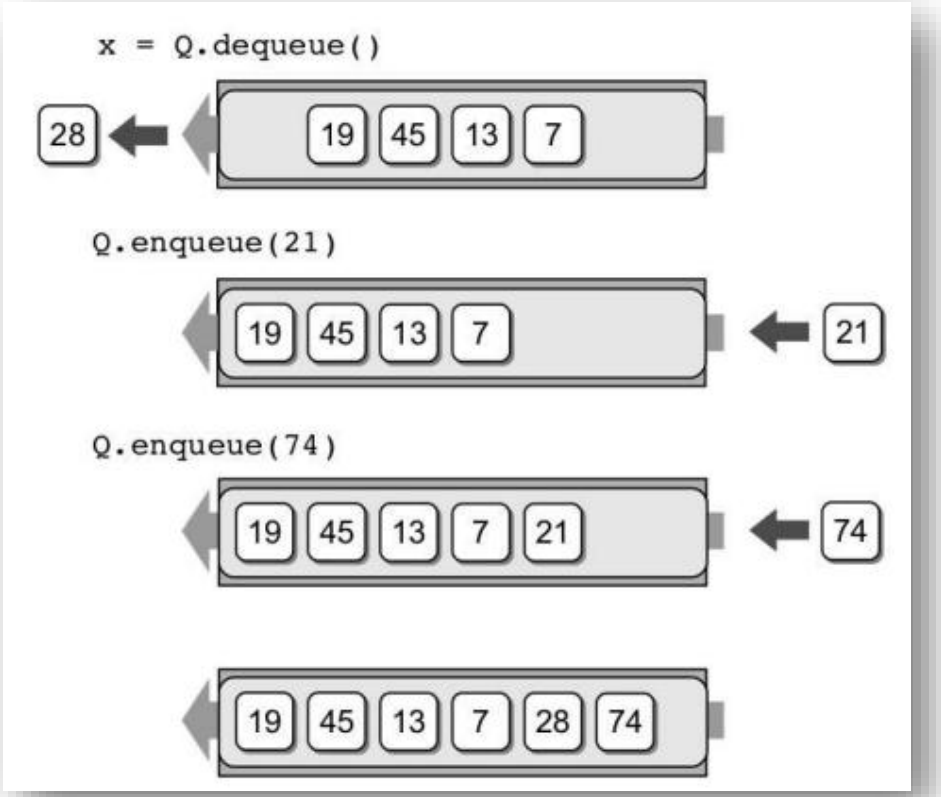
Operação	Conteúdo	Retorno	Descrição
q.is_empty()	[]	True	Verifica se fila está vazia
q.enqueue(4)	[4]		Insere elemento no final
q.enqueue('dog')	['dog', 4]		Insere elemento no final
q.enqueue(True)	[True, 'dog', 4]		Insere elemento no final
q.size()	[True, 'dog', 4]	3	Retorna número de elementos da fila
q.is_empty()	[True, 'dog', 4]	False	Verifica se fila está vazia
q.enqueue(8.4)	[8.4, True, 'dog', 4]		Insere elemento no final
q.dequeue()	[8.4, True, 'dog']	4	Remove elemento do início
q.dequeue()	[8.4, True]	'dog'	Remove elemento do início
q.size()	[8.4, True]	2	Retorna número de elementos da fila

FILAS

Operações básicas

Link: https://www.cs.usfca.edu/~galles/visualization/QueueArray.html





Filas

```
# Implementação da classe Fila
class Queue:
    # Inicia com uma fila vazia
    def __init__(self):
        self.itens = []

    # Verifica se fila está vazia
    def is_empty(self):
      return self.itens == []

    # Adiciona elemento no final da fila
    def enqueue(self, item):
        self.itens.insert(0, item)
        print(f'ENQUEUE {item}')

    # Remove elemento do inicio da fila
    def dequeue(self):
        print('DEQUEUE')
        return self.itens.pop()

    # Retorna o número de elementos da fila
    def size(self):
        return len(self.itens)

    # Imprime a fila na tela
    def print_queue(self):
        print(self.itens)
```

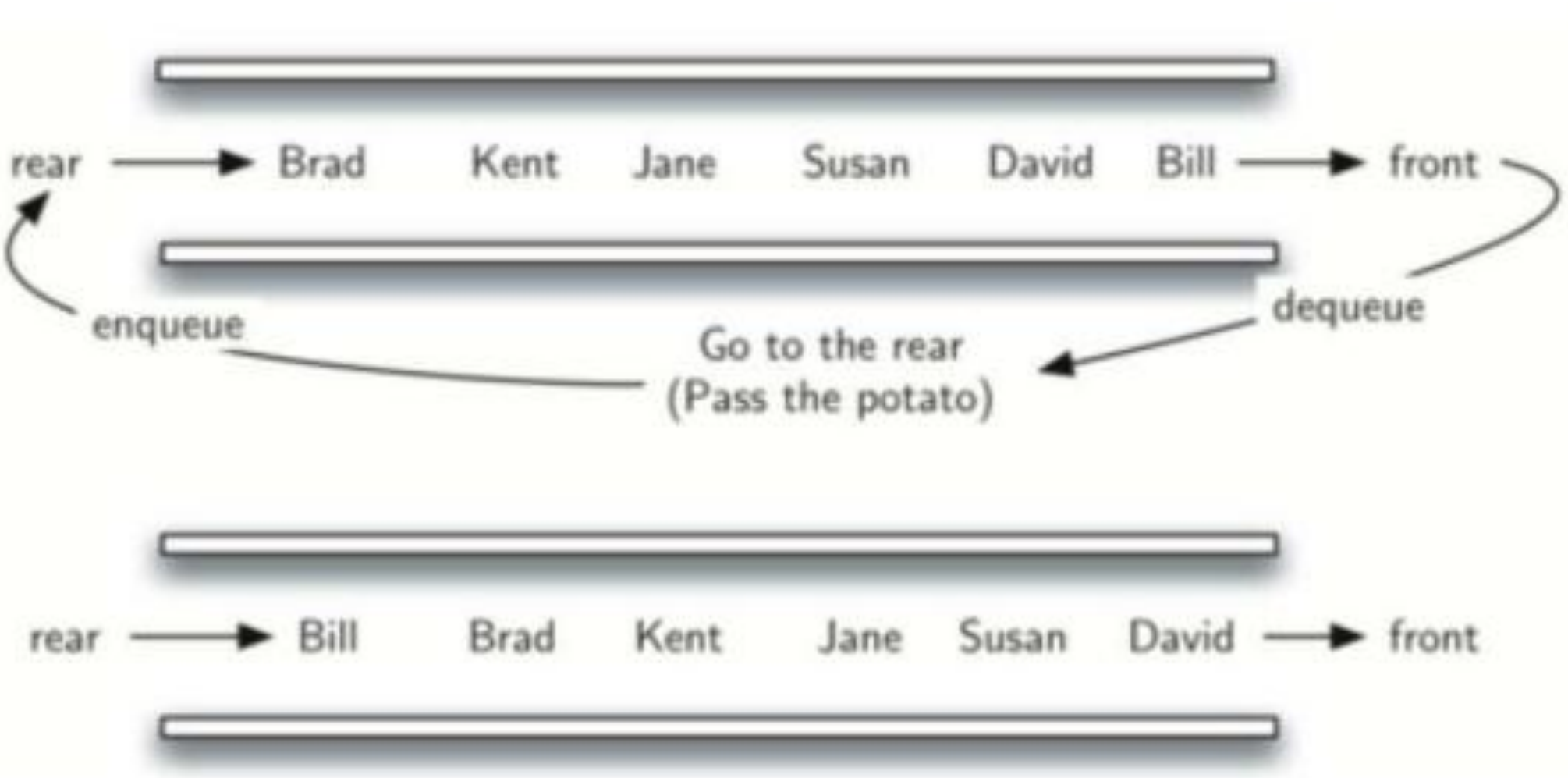
```
# TESTANDO
Q = Queue()
Q.print_queue()
Q.enqueue(1)
Q.enqueue(2)
Q.enqueue(3)
Q.print_queue()
Q.dequeue()
Q.dequeue()
Q.print_queue()
Q.enqueue(7)
Q.enqueue(8)
Q.enqueue(9)
Q.print_queue()
print(Q.is_empty())
```

VAMOS PARA A PRÁTICA ?!!!

APLICAÇÃO DE FILAS
Exercício 1: JOGO: Batata Quente

- Neste jogo, as crianças formam um círculo e passam um item qualquer (batata) cada um para o seu vizinho da frente o mais rápido possível.
- Em um certo momento do jogo, essa ação é interrompida (queimou) e a criança que estiver com o item (batata) na mão é excluída da roda.
- O jogo então prossegue até que reste apenas uma única criança, que é a vencedora.
- Para simular um círculo (roda), utilizaremos uma fila da seguinte maneira: a criança que está com a batata na mão será sempre a quela que estiver no início da fila. Após passar a batata, a simulação deve instantaneamente remover e inserir a criança, colocando-a novamente no final da fila. Ela então vai esperar até que todas as outras assumam o início da fila, antes de assumir essa posição novamente.
- Após um número pré estabelecido MAX de operações enqueue/dequeue, a criança que ocupar o início da fila será removida e outro ciclo da brincadeira é realizado.
- O processo continua até que a fila tenha possua tamanho um.

APLICAÇÃO DE FILAS
JOGO: Batata Quente



Batata Quente

```
# Utilização da classe QUEUE.
# Simula o jogo batata_quente
def batata_quente(nomes, MAX):
    # Cria fila para simular roda
    fila = Queue()

    # Coloca os N nomes em cada posição
    for nome in nomes:
        fila.enqueue(nome)

    # Inicia a lógica do jogo
    while fila.size() > 1:
        # Para simular MAX passagens da batata
        for i in range(MAX):
            # Remove o primeiro e coloca no final
            fila.enqueue(fila.dequeue())

        # Quem parar no ínicio da fila, está com batata
        # Deve ser eliminado da fila
        fila.dequeue()

    # Após N-1 rodadas, retorna a fila com o vencedor
    vencedor = fila.dequeue()
```

```
# TESTANDO
v = batata_quente(['Alex','Julia','Carlos','Maria','Ana','Caio'], 7)
print(f‘O vencedor é {v}’)
```

Exercícios

2. (ENADE) Em uma fila (estrutura de dados), é correto afirmar que:

A. O último elemento que entra é o primeiro a sair.

B. Não é possível inserir elementos enquanto realizamos remoções.

C. A primeira operação de enfileiramento insere no início da fila e a primeira de desenfileiramento remove do final.

D. O elemento removido numa operação dequeue é sempre o mais antigo (o primeiro a ter sido enfileirado).

E. Filas são acessadas apenas pelo meio da estrutura.

Exercícios

2. (ENADE) Em uma fila (estrutura de dados), é correto afirmar que:

A. O último elemento que entra é o primeiro a sair.

B. Não é possível inserir elementos enquanto realizamos remoções.

C. A primeira operação de enfileiramento insere no início da fila e a primeira de desenfileiramento remove do final.

D. O elemento removido numa operação dequeue é sempre o mais antigo (o primeiro a ter sido enfileirado).

E. Filas são acessadas apenas pelo meio da estrutura.

Exercícios

3. (ENADE) Considere as seguintes sequências de operações em uma fila vazia: enqueue(10), enqueue(20), dequeue(), enqueue(30), peek(). Qual será o valor retornado pela operação peek() (supondo que ela retorna o elemento da frente sem remover)?

A. 10
B. 20
C. 30
D. Nenhum, pois a fila está vazia.
E. Depende da implementação da fila.

Exercícios

3. (ENADE) Considere as seguintes sequências de operações em uma fila vazia: enqueue(10), enqueue(20), dequeue(), enqueue(30), peek(). Qual será o valor retornado pela operação peek() (supondo que ela retorna o elemento da frente sem remover)?

A. 10
B. 20
C. 30
D. Nenhum, pois a fila está vazia.
E. Depende da implementação da fila.

Exercícios

4. (ENADE) Em qual situação faz mais sentido usar uma fila (FIFO) ao invés de uma pilha (LIFO)?

A. Implementar as operações “desfazer” de um editor de texto (undo).
B. Organizar a sequência de páginas visitadas por um navegador para voltar (back).
C. Controle de execução de tarefas sequenciais que chegam e saem na ordem exata de chegada.
D. Desempilhar livros de uma pilha empilhada em cima de uma mesa.
E. Carregar contêineres em ordem inversa.

Exercícios

4. (ENADE) Em qual situação faz mais sentido usar uma fila (FIFO) ao invés de uma pilha (LIFO)?

A. Implementar as operações “desfazer” de um editor de texto (undo).
B. Organizar a sequência de páginas visitadas por um navegador para voltar (back).
C. Controle de execução de tarefas sequenciais que chegam e saem na ordem exata de chegada.
D. Desempilhar livros de uma pilha empilhada em cima de uma mesa.
E. Carregar contêineres em ordem inversa.

APLICAÇÃO DE FILAS
Exercício 5: ATENDIMENTO EM UM BANCO

- Simule uma fila de atendimento em um banco, onde as pessoas são chamadas na ordem em que chegaram. O próximo cliente da fila é atendido e removido da fila. A cada três atendimentos, o atendente faz uma pausa, e novos clientes podem ser adicionados à fila durante esse período.
- **Requisitos:**

1. Criar a fila com os nomes de clientes.
2. Processar três atendimentos e depois fazer uma pausa.
3. Durante a pausa, adicionar dois novos clientes à fila.
4. Continuar até que todos os clientes tenham sido atendidos.

Atendimento em um Banco

```
# Utilização da classe Queue para o atendimento no banco
def atendimento_banco(clientes):
    fila = Queue()

    # Coloca os clientes iniciais na fila
    for cliente in clientes:
        fila.enqueue(cliente)

    atendimento = 0
    while not fila.is_empty():
        if atendimento == 3:
            print("PAUSA: Adicionando novos clientes.")
            fila.enqueue("Novo Cliente 1")
            fila.enqueue("Novo Cliente 2")
            atendimento = 0  # Resetando o contador de atendimentos

        # Atendendo o próximo cliente da fila
        print(f"Atendendo: {fila.dequeue()}")
        atendimento += 1
```

```
# Testando
clientes_iniciais = ['Carlos', 'Maria', 'Ana', 'João', 'Paula']
atendimento_banco(clientes_iniciais)
```

APLICAÇÃO DE FILAS
Exercício 6: CONTROLE DE PEDIDOS EM UM RESTAURANTE

- Implemente um sistema de controle de pedidos em um restaurante. Os pedidos são colocados em uma fila conforme são recebidos. A cada certo número de minutos, o sistema processa um pedido (remove o primeiro pedido da fila) e adiciona um novo pedido. Continue o processo até que uma condição de parada (número total de pedidos processados) seja atingida.
- **Requisitos:**

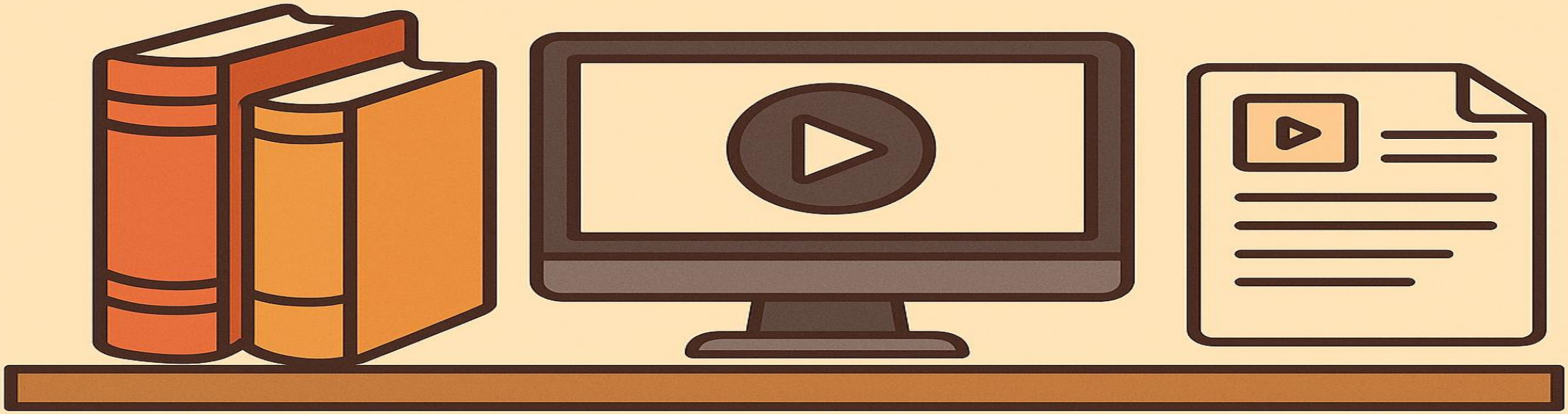
1. Colocar pedidos na fila.
2. Processar um pedido a cada intervalo e adicionar um novo pedido.
3. Parar o processo após processar um número determinado de pedidos.

Controle de Pedidos em um Restaurante

```
# Utilização da classe Queue para controle de pedidos
def controle_pedidos(pedidos, num_processados):
    fila = Queue()

    # Adicionando os pedidos iniciais na fila
    for pedido in pedidos:
        fila.enqueue(pedido)

    pedidos_processados = 0

    # Processamento de pedidos
    while pedidos_processados < num_processados:
        # Processa o pedido no início da fila
        print(f"Processando pedido: {fila.dequeue()}")
        pedidos_processados += 1

        # Adicionando um novo pedido
        novo_pedido = f"Pedido {pedidos_processados + len(pedidos)}"
        fila.enqueue(novo_pedido)
        print(f"Novo pedido adicionado: {novo_pedido}")
```

```
# Testando
pedidos_iniciais = ['Pedido 1', 'Pedido 2', 'Pedido 3']
controle_pedidos(pedidos_iniciais, 5)
```

Próxima Aula



- Ler o capítulo 8 do livro “Estrutura de Dados com Python”

Boa semana e bons estudos!!