

Algoritmos e Estruturas de Dados

Aula 3

Prof Dr Tanilson Dias dos Santos

Universidade Aberta do Brasil – UAB
Universidade Federal do Tocantins - UFT



Da Aula Anterior...

- **Conceito de Pilha:**
 - **Política Associada à Estrutura de Dados Pilha;**
 - **Algumas Aplicações.**
- **Prática de Programação:**
 - **Pilhas em Python;**
 - **Listas como Pilhas.**



Na Aula de Hoje, Veremos...

- **Conceito de Fila:**
 - **Política Associada à Estrutura de Dados Fila;**
 - **Algumas Aplicações.**
- **Prática de Programação:**
 - **Filas em Python;**
 - **Listas como Filas.**

O que são Filas?



- Fila é uma estrutura de dados que encontramos com alguma frequência no mundo real;
- A fila consiste em uma estrutura onde você insere elementos em uma extremidade, e na hora de retirar, então retira-se os elementos da outra extremidade;
- Isso define uma política.

O que são filas?



Fila

- A política ou princípio que está relacionada à Fila é a FIFO;
- **FIFO – First in, First Out**
- Em bom português, dizemos que "O primeiro a entrar é o primeiro a Sair".
- Assim como em uma fila do mundo real, onde as pessoas são atendidas na ordem em que chegaram, a fila em Python mantém essa mesma lógica.

Onde Vemos Filas?



Fila de
Atendimento em
um Call Center



Fila de Supermercado

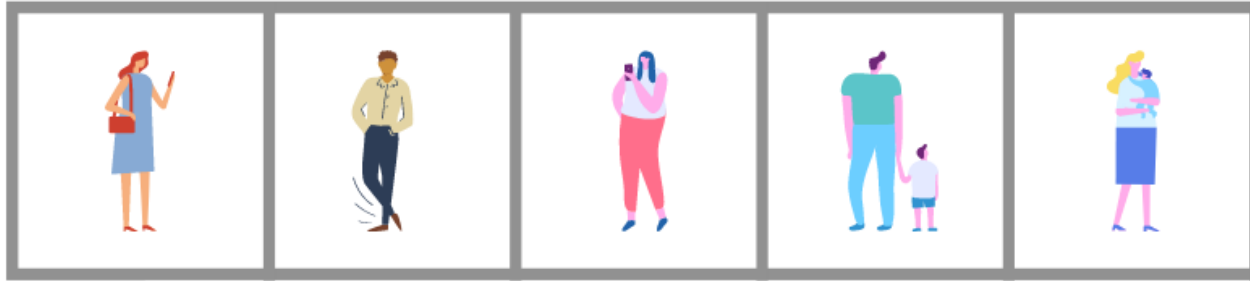


Fila de Prioridades

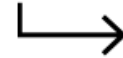
Funcionamento da Estrutura

Fim

Início

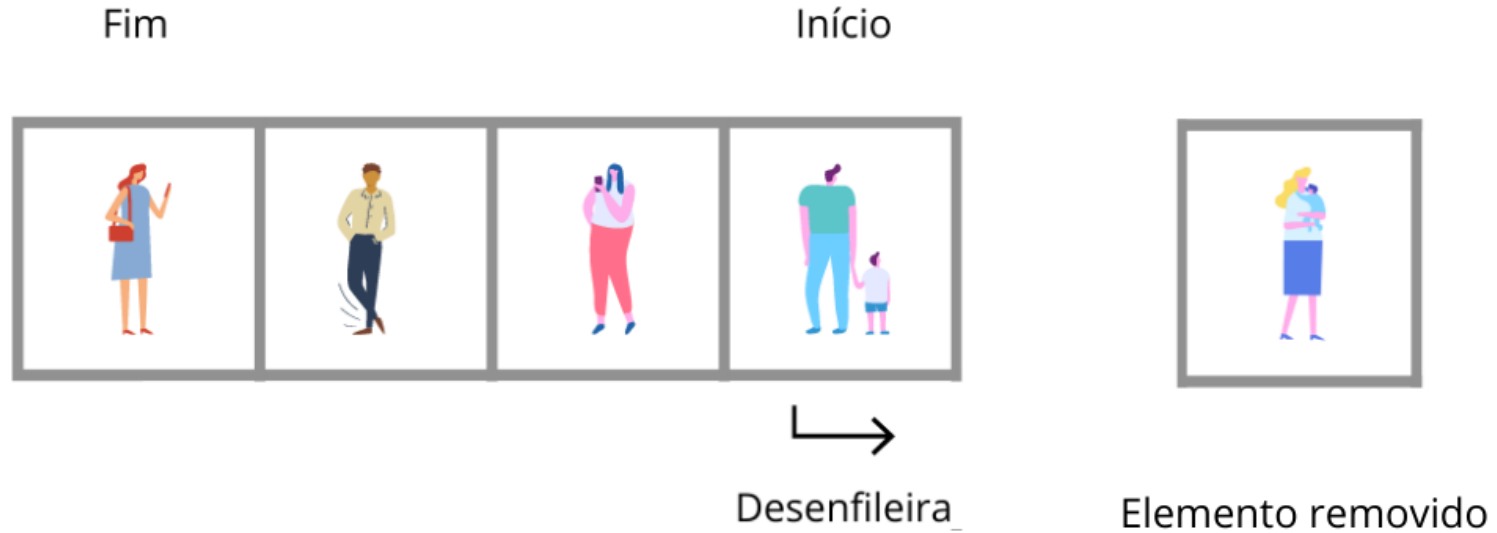


- As **inserções** na Fila ocorrem sempre no **final** da fila;
- Analogamente, as **remoções** ocorrem sempre no **início** da fila.



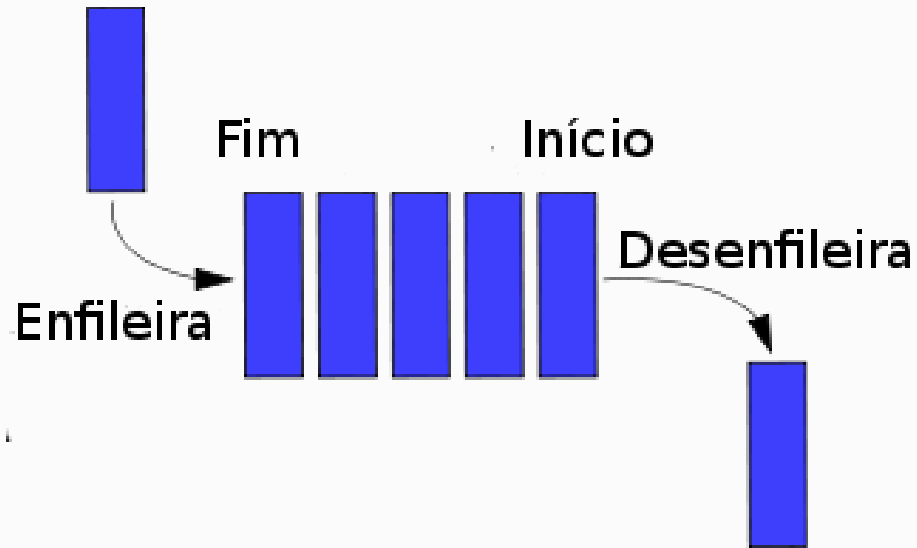
Desenfileira

Funcionamento da Estrutura



- As remoções ocorrem sempre no início da fila.

Funcionamento da Estrutura



Implementando Filas

- Em Python já existe uma biblioteca do tipo Fila (**Queue**);
- Também é possível criar a nossa própria classe Fila;
- Vamos ver como utilizar a classe Fila das duas formas a seguir.

Implementando Filas

- A função "**put()**" serve para inserir elementos na fila.
- A função "**get()**" serve para remover elementos na fila.
- Quem é a nova cabeça da Fila ao final da execução do código ao lado?

```
from queue import Queue
```

```
fila = Queue()
```

```
# Adicionando elementos na fila
```

```
fila.put(10)
```

```
fila.put(20)
```

```
fila.put(30)
```

```
# Removendo elementos da fila
```

```
elemento = fila.get()
```

```
print("elemento:", elemento)
```

Implementando Filas

- E se retirarmos elementos de uma Fila e inserirmos em outra?
- Como ficam as Filas fila1 e fila2 ao final da execução do código a seguir?
- Faça o print das duas filas para ver o que acontece.

```
print ( fila1 )  
print( fila2 )
```

```
from queue import Queue
```

```
fila1 = Queue()
```

```
fila2 = Queue()
```

```
fila1.put(10)
```

```
fila1.put(20)
```

```
fila2.put(30)
```

```
fila2.put(fila1.get())
```

```
fila1.put(8)
```

```
fila1.put(fila2.get())
```

```
fila2.put(15)
```

Implementando Filas

- E se retirarmos elementos de uma Fila e inserirmos em outra?
- Como ficam as Filas fila1 e fila2 ao final da execução do código a seguir?

```
def mostrar(fila):  
    for elem in range(fila.qsize()):  
        print(fila.get())
```

```
from queue import Queue
```

```
fila1 = Queue()
```

```
fila2 = Queue()
```

```
fila1.put(10)
```

```
fila1.put(20)
```

```
fila2.put(30)
```

```
fila2.put(fila1.get())
```

```
fila1.put(8)
```

```
fila1.put(fila2.get())
```

```
fila2.put(15)
```

Implementando Filas

- E se retirarmos elementos de uma Fila e inserirmos em outra?
- Como ficam as Filas fila1 e fila2 ao final da execução do código a seguir?

```
mostrar(fila1)
```

```
mostrar(fila1)
```

```
from queue import Queue
```

```
fila1 = Queue()
```

```
fila2 = Queue()
```

```
fila1.put(10)
```

```
fila1.put(20)
```

```
fila2.put(30)
```

```
fila2.put(fila1.get())
```

```
fila1.put(8)
```

```
fila1.put(fila2.get())
```

```
fila2.put(15)
```

Implementando Filas

- E se retirarmos elementos de uma Fila e inserirmos em outra?
- Como ficam as Filas fila1 e fila2 ao final da execução do código a seguir?

```
def mostrarSemPerder(fila):  
    tamanho = fila.qsize()  
    for elem in range(tamanho):  
        aux = fila.get()  
        print(aux)  
        fila.put(aux)
```

```
from queue import Queue
```

```
fila1 = Queue()
```

```
fila2 = Queue()
```

```
fila1.put(10)
```

```
fila1.put(20)
```

```
fila2.put(30)
```

```
fila2.put(fila1.get())
```

```
fila1.put(8)
```

```
fila1.put(fila2.get())
```

```
fila2.put(15)
```

Por que Utilizar Filas?

As filas em Python são amplamente utilizadas em projetos de tecnologia devido às diversas vantagens que oferecem:

- **Ordem de processamento** (fila de processamento do sistema operacional): As filas garantem que as tarefas sejam processadas na ordem em que foram adicionadas. Isso é especialmente útil em sistemas que processam grandes quantidades de dados ou executam tarefas em paralelo.

Por que Utilizar Filas?

- **Controle de fluxo** (evitar gargalos): Ao utilizar filas, é possível controlar o fluxo de informações ou requisições em um sistema. Por exemplo, em um servidor web, as requisições podem ser adicionadas em uma fila e processadas uma por uma, evitando sobrecarga e garantindo um melhor desempenho.
- **Sincronização de threads** (recursos compartilhados): Quando lidamos com programação concorrente e múltiplas threads, as filas oferecem uma maneira segura de compartilhar recursos entre as threads. As threads podem adicionar e remover elementos da fila de forma sincronizada, evitando problemas de concorrência.

Definindo Nossa Classe Fila

- Vamos utilizar Listas como Filas!
- Vamos utilizar a função **append()** para inserir elementos ao final da fila;
- Vamos usar a função **pop()** para remover elementos do início da fila.

```
class Fila:
```

```
    def __init__(self):  
        self.fila = [ ]
```

```
    def adicionar(self, elemento):  
        self.fila.append(elemento)
```

```
    def remover(self):  
        if not self.vazia():  
            return self.fila.pop(0)
```

```
    def estaVazia(self):  
        return len(self.fila) == 0
```

Listas como Filas

- Instanciem o objeto;
- Adicionem elementos e escrevam um método para mostrar a fila.

```
fila = Fila()
```

```
# Adicionando elementos na fila
```

```
fila.adicionar(18)
```

```
fila.adicionar(21)
```

```
fila.adicionar(17)
```

```
class Fila:
```

```
    def __init__(self):  
        self.fila = [ ]
```

```
    def adicionar(self, elemento):  
        self.fila.append(elemento)
```

```
    def remover(self):  
        if not self.vazia():  
            return self.fila.pop(0)
```

```
    def estaVazia(self):  
        return len(self.fila) == 0
```

Listas como Filas

Cabeça



Cauda

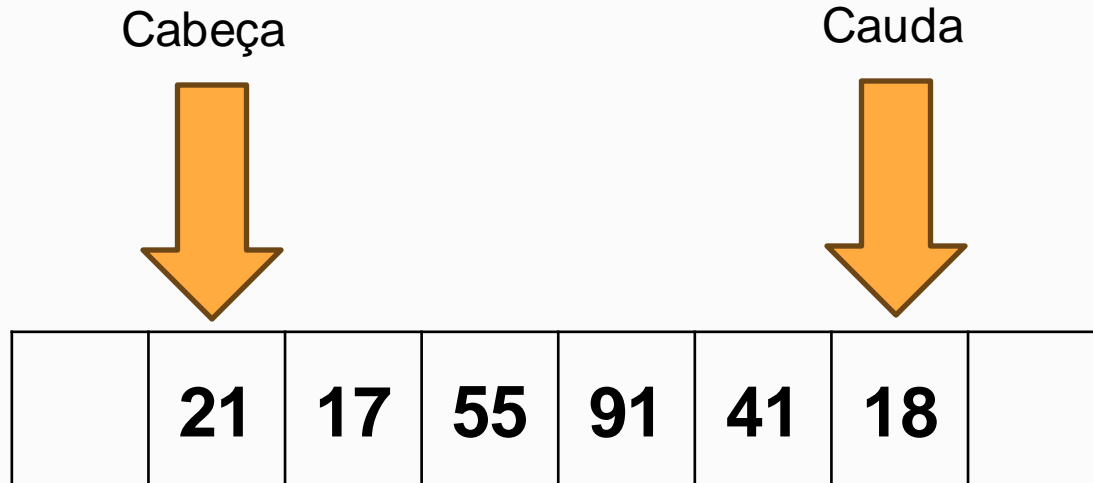


18	21	17					
----	----	----	--	--	--	--	--

```
fila.adicionar(55)  
fila.adicionar(91)  
elemento = fila.remover()  
fila.adicionar(41)  
fila.adicionar(elemento)
```

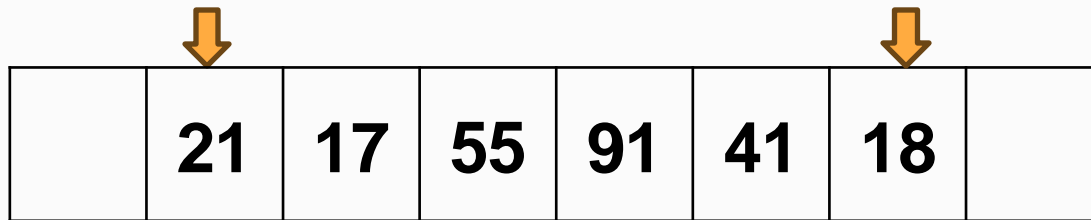
Listas como Filas

- Modifiquem a função de **mostrar a fila**. A Saída deve ser algo dessa forma:



Listas como Filas

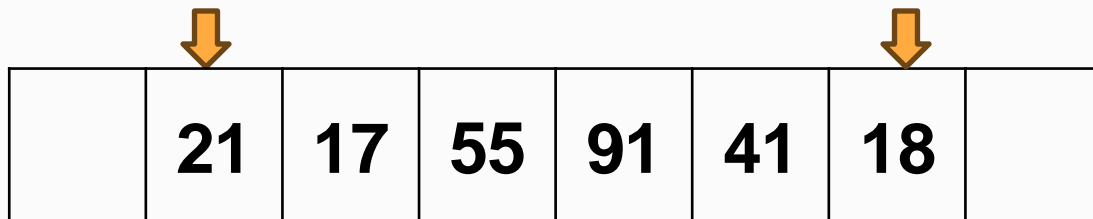
```
def mostrarSemPerder(self):  
    tamanho = len(self.fila)  
    for elem in range(tamanho):  
        aux = self.fila.pop(0)  
        print(aux)  
        self.fila.append(aux)
```



Exercícios com Filas (1)

- Considere as filas f1 e f2, inicialmente como descritas abaixo. Como transferir todos os elementos de f1 para f2 **utilizando somente as funções da Classe Fila que criamos?**
- Imprimir as duas filas ao final.

Fila f1 =



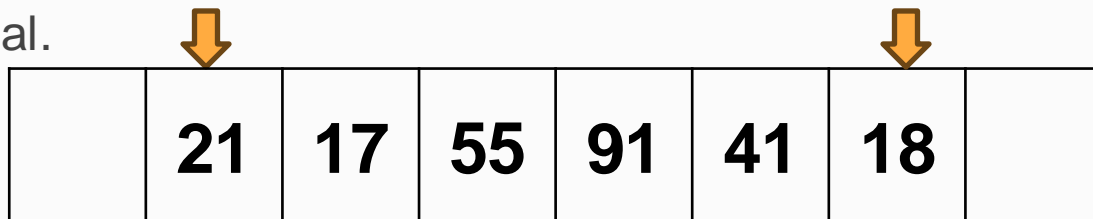
Fila f2 =



Exercícios com Filas (2)

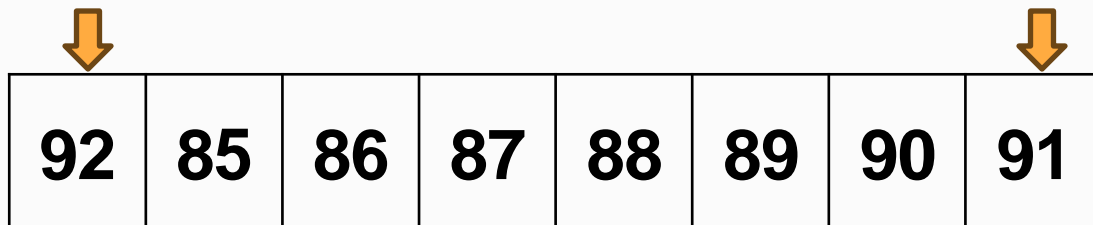
- Considere as filas f1 e f2, inicialmente como descritas abaixo. Criar uma fila f3 que seja a concatenação dos elementos de f1 seguidos dos elementos de f2, **utilizando somente as funções da Classe Fila que criamos**.
- Imprimir f3 ao final.

Fila f1 =



	21	17	55	91	41	18	
--	----	----	----	----	----	----	--

Fila f2 =

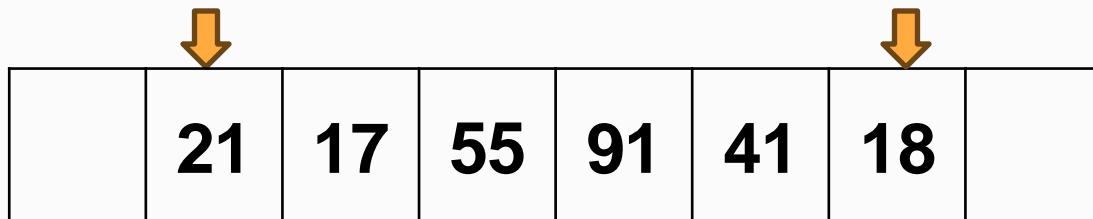


92	85	86	87	88	89	90	91
----	----	----	----	----	----	----	----

Exercícios com Filas (3)

- Considere as filas f1 e f2, inicialmente como descritas abaixo. Criar uma fila f3 que seja a intercalação dos elementos de f1 e f2, **utilizando somente as funções da Classe Fila que criamos**.
- Imprimir f3 ao final.

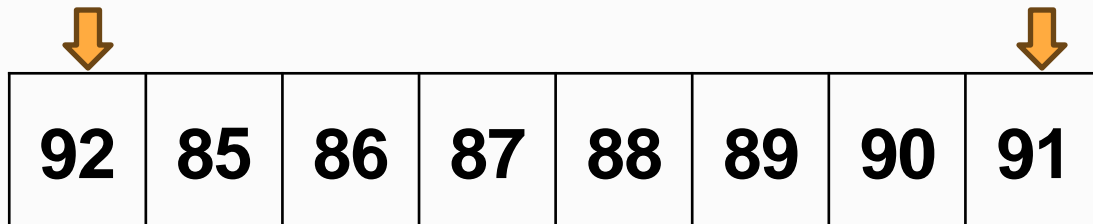
Fila f1 =



A horizontal array of 8 cells representing a queue. The second cell contains '21', the third '17', the fourth '55', the fifth '91', the sixth '41', and the seventh '18'. The first and eighth cells are empty. An orange arrow points down to the second cell, and another orange arrow points down to the seventh cell.

	21	17	55	91	41	18	
--	----	----	----	----	----	----	--

Fila f2 =



A horizontal array of 8 cells representing a queue. The first cell contains '92', the second '85', the third '86', the fourth '87', the fifth '88', the sixth '89', the seventh '90', and the eighth '91'. An orange arrow points down to the first cell, and another orange arrow points down to the eighth cell.

92	85	86	87	88	89	90	91
----	----	----	----	----	----	----	----

Resumo da Aula e Plano de Estudos



Tarefas Semanais

- Refazer Exercícios da Aula;
- Responder Questionário Avaliativo (vale 1.0 ponto);
- Responder Fórum;
- Fazer leitura recomendada;
- Revisar Listas, Pilhas e Filas em Python;
- Próxima Aula vamos ver Notações de Complexidade e encadeamento de Pilhas e Filas.



Conclusão e Próxima Aula

- **Aula de Hoje:**
 - Apresentação principais conceitos de Fila;
 - Tipos Abstratos de Dados e Manipulação de Filas;
 - Prática de Programação.
- **Próxima Aula:**
 - Fias em Python;
 - Listas como Filas.

Algoritmos e Estruturas de Dados

Aula 3

Prof Dr Tanilson Dias dos Santos

Universidade Aberta do Brasil – UAB
Universidade Federal do Tocantins - UFT