# Educação Profissional Paulista

## Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

# Lógica de programação e algoritmos

## Pilhas e filas

Aula 2

Código da aula: [DADOS]ANO1C3B4S28A2

# Mapa da Unidade 1 Componente 4

**Lógica de programação e algoritmos**

semana 21

Recursividade

semana 23

Busca e ordenação

semana 25

Prática de busca e ordenação

semana 27

Algoritmos de contagem e acumulação

semana 28

**Você está aqui!**

Pilhas e filas

Lógica de programação e algoritmos

Mapa da Unidade 1 Componente 4

# Você está aqui!

28

Pilhas e filas

## Aula 2

Código da aula:
[DADOS]ANO1C3B4S28A2

## Objetivos da aula

- Introduzir os fundamentos das estruturas de dados pilhas e filas.

## Recursos didáticos

- Recurso audiovisual para exibição de vídeos e imagens;
- Acesso ao laboratório de informática e/ou internet.

## Duração da aula

50 minutos.

## Competências técnicas

- Identificar e resolver problemas relacionados a dados e análises;
- Compreender e dominar técnicas de manipulação de dados.

## Competências socioemocionais

- Adaptar-se a novas tecnologias, técnicas e tendências sem perder o foco, as metas e os objetivos da organização;
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais colaborando com colegas, gestores e clientes.

Construindo o **conceito**

# Implementação de pilhas usando listas e deques

Em Python, podemos implementar pilhas usando tanto listas quanto deques (da biblioteca **collections**).

As listas fornecem uma implementação básica, enquanto os deques oferecem uma performance mais eficiente para operações de push e pop devido à sua natureza de lista duplamente encadeada.

Construindo
o **conceito**

# Implementação de pilhas usando listas e deques

Similarmente, as filas podem ser implementadas usando listas e deques.

Enquanto as listas são simples de usar, os deques são mais eficientes para operações de enqueue e dequeue, oferecendo tempo constante O(1) para essas operações.

Construindo o **conceito**

# Aplicações avançadas de pilhas

### Recursão e pilhas de execução

Pilhas são usadas para gerenciar chamadas recursivas e a pilha de execução de um programa.

### Algoritmos de busca em grafos

DFS (Depth first search) utiliza pilhas para explorar caminhos.

### *Parsing* de linguagens

Analisadores sintáticos utilizam pilhas para gerenciar a estrutura da gramática.

Construindo o **conceito**

# Aplicações avançadas de pilhas

### Algoritmos de busca em grafos

BFS (*Breadth first search*) utiliza filas para explorar níveis de nós.

### Sistemas de mensagens assíncronas

Filas são usadas para gerenciar mensagens entre processos.

### Simulação de filas de atendimento

Filas são usadas para modelar e simular sistemas de atendimento, como filas em bancos e call centers.

Construindo o **conceito**

# Comparação entre listas e deques para pilhas e filas

- **Listas:** Simples de usar, mas ineficientes para operações de inserção e remoção em grandes volumes de dados.

- **Deques:** Mais eficientes, oferecendo tempo constante para inserções e remoções tanto no início quanto no fim da estrutura.

Construindo o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de pilhas usando listas e deques.

**Exemplo 1:** pilha usando lista:

```
class StackList:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop() if self.items else None

    def peek(self):
        return self.items[-1] if self.items else None

    def is_empty(self):
        return not self.items
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo
o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de pilhas usando listas e deques.

**Exemplo 1:** pilha usando lista:

```
# Exemplo de uso
stack_list = StackList()
stack_list.push(1)
stack_list.push(2)
stack_list.push(3)
print("Pilha usando lista:", stack_list.items)
print("Pop:", stack_list.pop())
print("Peek:", stack_list.peek())
```

Construindo
o **conceito**

# Exemplos práticos

## Vamos praticar!

Implementação de pilhas usando listas e deques.

**Exemplo 2:** pilha usando deque:

```python
from collections import deque

class StackDeque:
    def __init__(self):
        self.items = deque()

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop() if self.items else None

    def peek(self):
        return self.items[-1] if self.items else None

    def is_empty(self):
        return not self.items
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de pilhas usando listas e deques.

**Exemplo 2:** pilha usando deque:

```
# Exemplo de uso
stack_deque = StackDeque()
stack_deque.push(1)
stack_deque.push(2)
stack_deque.push(3)
print("Pilha usando deque:", stack_deque.items)
print("Pop:", stack_deque.pop())
print("Peek:", stack_deque.peek())
```

Construindo o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de filas usando listas e deques.

**Exemplo 3:** fila usando lista:

```python
class QueueList:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        return self.items.pop(0) if self.items else None

    def front(self):
        return self.items[0] if self.items else None

    def is_empty(self):
        return not self.items
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo
o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de filas usando listas e deques.

**Exemplo 3:** fila usando lista:

```
# Exemplo de uso
queue_list = QueueList()
queue_list.enqueue(1)
queue_list.enqueue(2)
queue_list.enqueue(3)
print("Fila usando lista:", queue_list.items)
print("Dequeue:", queue_list.dequeue())
print("Front:", queue_list.front())
```

Construindo o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de filas usando listas e deques.

**Exemplo 4:** fila usando deque:

```python
from collections import deque

class QueueDeque:
    def __init__(self):
        self.items = deque()

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        return self.items.popleft() if self.items else None

    def front(self):
        return self.items[0] if self.items else None

    def is_empty(self):
        return not self.items
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Construindo
o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de filas usando listas e deques.

**Exemplo 4:** fila usando deque:

```
# Exemplo de uso
queue_deque = QueueDeque()
queue_deque.enqueue(1)
queue_deque.enqueue(2)
queue_deque.enqueue(3)
print("Fila usando deque:", queue_deque.items)
print("Dequeue:", queue_deque.dequeue())
print("Front:", queue_deque.front())
```

Construindo o **conceito**

# Exemplos práticos

**Vamos praticar!**

Implementação de filas usando listas e deques.

**Exemplo 5:** outros exemplos:

**Pilha usando lista:**

```
stack = []
stack.append(1)  # push
stack.append(2)
stack.append(3)
print("Pilha após pushes:", stack)
print("Pop:", stack.pop())  # pop
print("Peek:", stack[-1])  # peek
print("Pilha final:", stack)
```

**Pilha usando deque:**

```
from collections import deque
stack = deque()
stack.append(1)  # push
stack.append(2)
stack.append(3)
print("Pilha após pushes:", stack)
print("Pop:", stack.pop())  # pop
print("Peek:", stack[-1])  # peek
print("Pilha final:", stack)
```

Construindo o **conceito**

# Exemplos práticos

## Vamos praticar!

Implementação de filas usando listas e deques.

**Exemplo 5:** outros exemplos:

### Pilha usando lista:

```
queue = []
queue.append(1)  # enqueue
queue.append(2)
queue.append(3)
print("Fila após enqueues:", queue)
print("Dequeue:", queue.pop(0))  # dequeue
print("Front:", queue[0])  # front
print("Fila final:", queue)
```

### Pilha usando deque:

```
from collections import deque
queue = deque()
queue.append(1)  # enqueue
queue.append(2)
queue.append(3)
print("Fila após enqueues:", queue)
print("Dequeue:", queue.popleft())  # dequeue
print("Front:", queue[0])  # front
print("Fila final:", queue)
```

Colocando em **prática**

# Exercício: manipulação de pilhas e filas

Trabalhe em dupla para implementar as classes stack e queue usando listas em Python.

Após a implementação, execute as operações indicadas e verifique se o comportamento das estruturas de dados está correto.

Compartilhe suas implementações e resultados com a classe.

**20 minutos**

**Duplas**

**Código de programação**

Colocando em **prática**

# Exercício: manipulação de pilhas e filas

**20 minutos**

**Duplas**

**Código de programação**

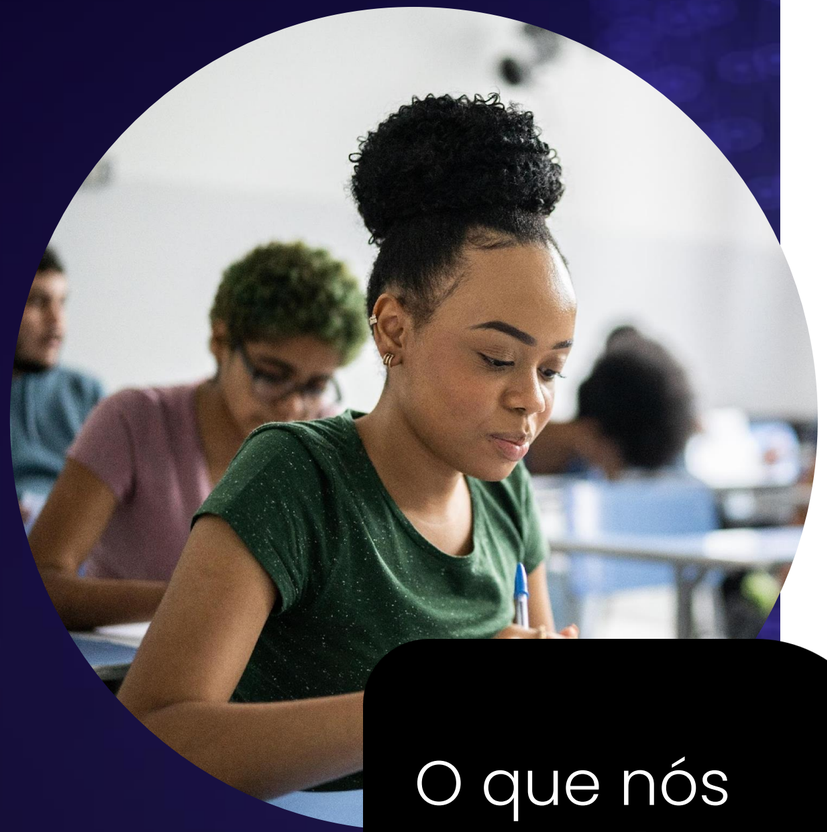
## IMPLEMENTAÇÃO DA PILHA

- Implemente a classe Stack com as seguintes operações: push, pop, peek, e is_empty.
- Teste sua implementação com a sequência de operações fornecida.

## IMPLEMENTAÇÃO DA FILA

- Implemente a classe queue com as seguintes operações: enqueue, dequeue, front, e is_empty.
- Teste sua implementação com a sequência de operações fornecida.

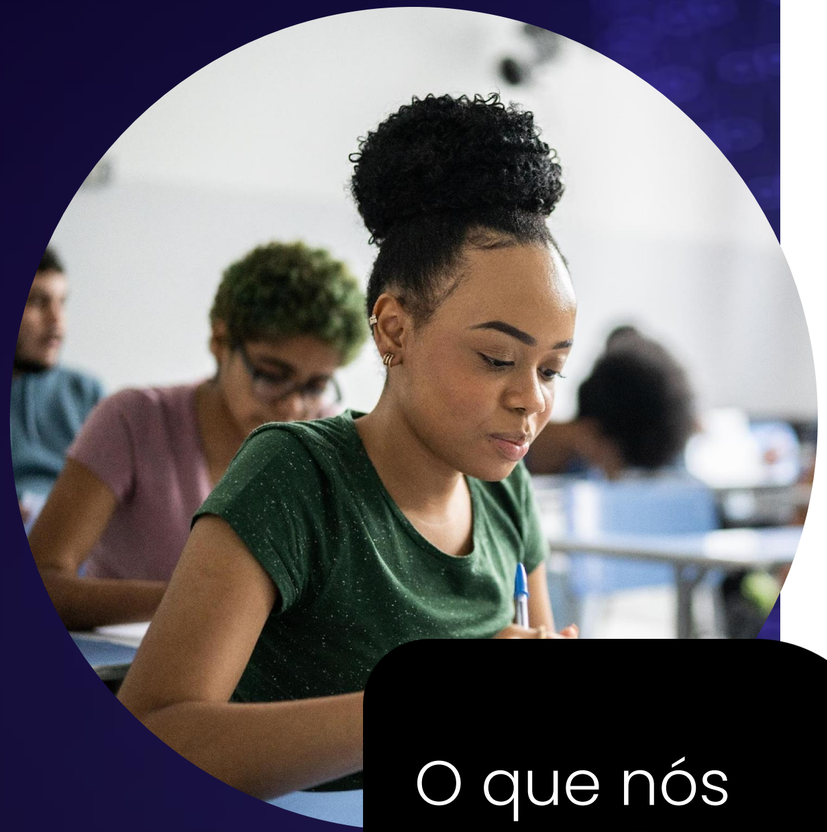
Diálogo



© Getty Images

O que nós **aprendemos hoje?**

# Nuvem de palavras





O que nós **aprendemos hoje?**

# Então ficamos assim...

1. Na segunda aula, aprofundamos a implementação de pilhas e filas utilizando listas e deques em Python. Exploramos como essas estruturas oferecem diferentes eficiências para operações básicas de inserção e remoção de elementos.

2. Demonstramos aplicações avançadas de pilhas, como a gestão de chamadas recursivas e algoritmos de busca em grafos. Também discutimos aplicações de filas em algoritmos como BFS e sistemas de mensagens assíncronas.

3. Concluímos com um exercício prático, no qual os alunos implementaram e testaram suas próprias versões de pilhas e filas, consolidando o entendimento das operações e aplicações dessas estruturas de dados.

# Saiba mais

Você ainda pode aprender muito mais!

Veja este curso e conheça Python tão bem quanto a palma da sua mão!

ALURA. Curso - Python para Data Science: Trabalhando com funções, estruturas de dados e exceções. Disponível em: https://www.alura.com.br/curso-online-python-data-science-funcoes-estruturas-dados-excecoes. Acesso em: 24 jul. 2024.

# Referências da aula

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. **Lógica de programação**: a construção de algoritmos e estruturas de dados com aplicações em Python. Porto Alegre: Bookman, 2022

# Educação Profissional Paulista

## Técnico em **Ciência de Dados**

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO