Estrutura de Dados

Profª. Drª. Alana Morais
alanamm.prof@gmail.com

Objetivo da aula

Entender o comportamento das pilhas

Recapitulando - Estrutura de dados

Dados simples:

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

Dados estruturados:

- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.

Recapitulando - Estrutura de dados

Dados simples:

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

Dados estruturados:

- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.

listas ou vetores

Recapitulando - Estrutura de dados

Dados simples:

- padrão:
  - inteiro (int);
  - real (float);
  - caracter (str);
  - lógico (boolean).

Dados estruturados:

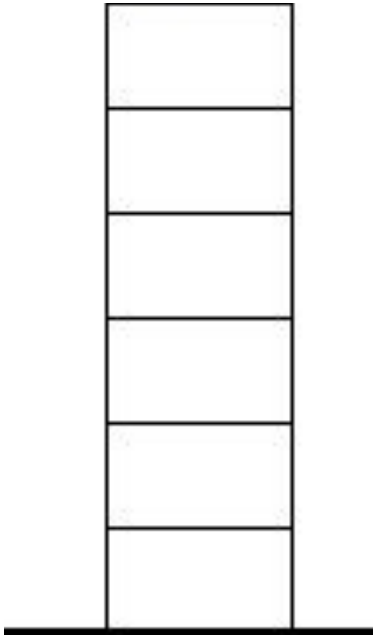
- Estáticos:
  - arrays;
  - registros;
  - arquivos;
  - conjuntos;
  - cadeias.
- Dinâmicos:
  - filas;
  - pilhas;
  - listas encadeadas;
  - árvores;
  - grafos.

listas ou vetores

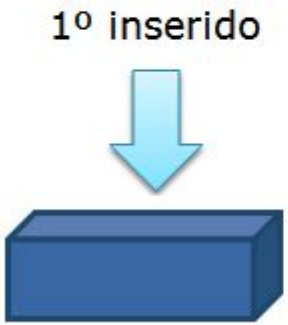
Pilhas

Pilha

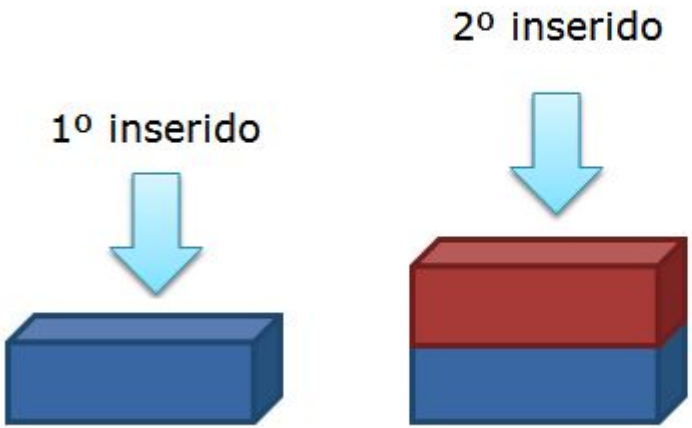
First In, Last Out



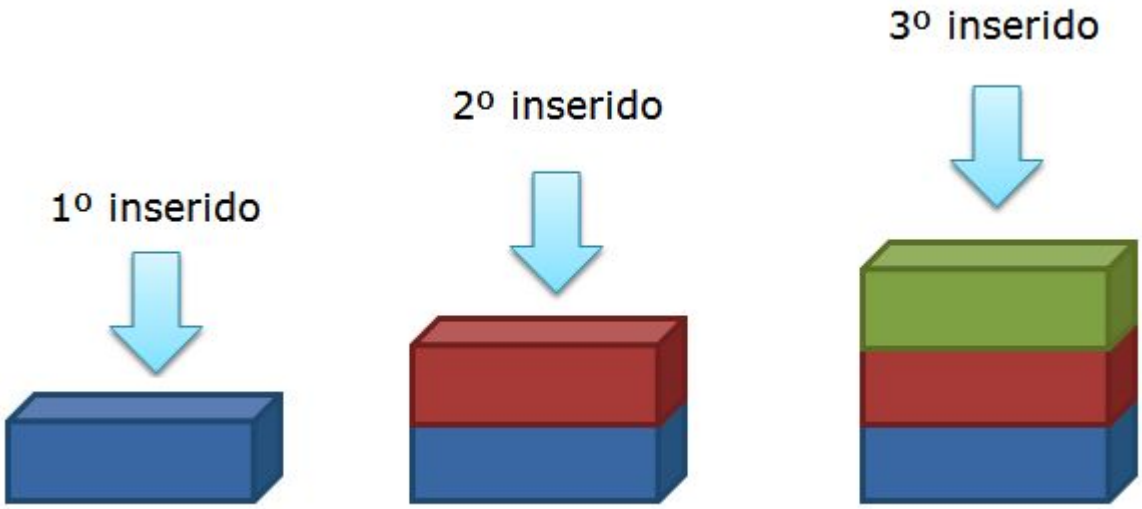
Pilhas (Inserção)



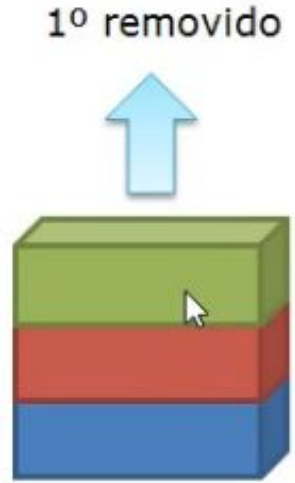
Pilhas (Inserção)



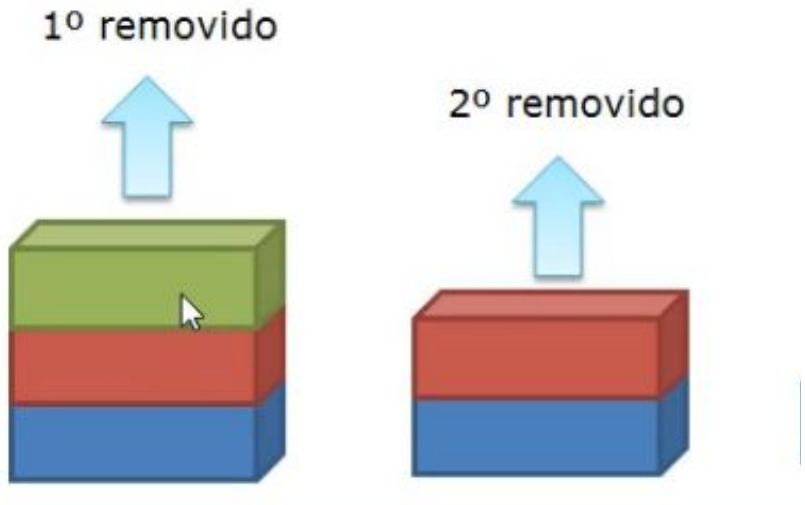
Pilhas (Inserção)



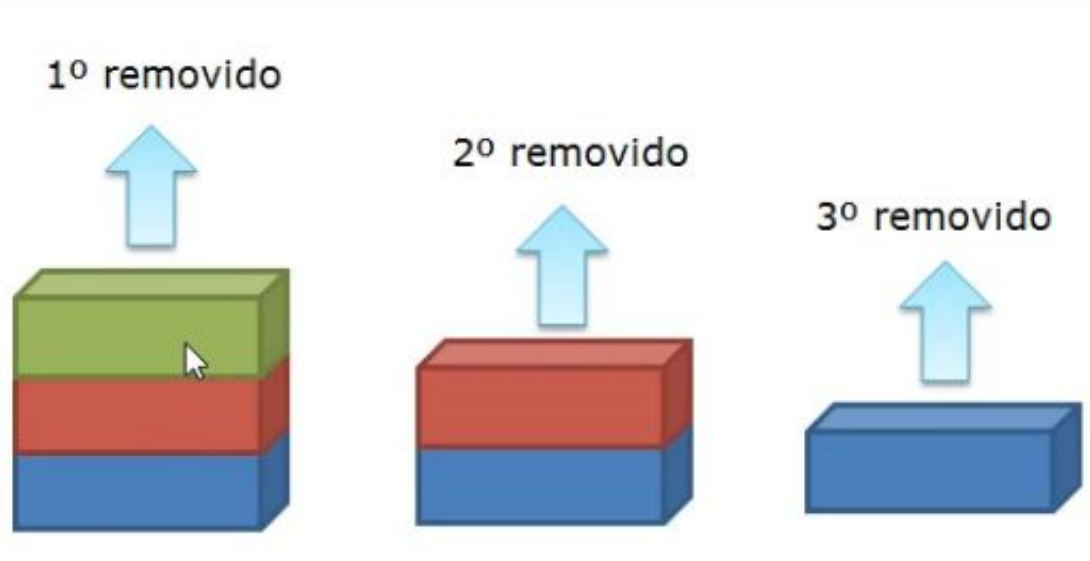
Pilhas (Remoção)



Pilhas (Remoção)



Pilhas (Remoção)



Pilhas - Operações

- Criação;
- Mostrar valores da pilha;
- Inserção de elementos na pilha;
- Remoção de elementos na pilha;
- Descoberta do elemento do topo;
- Esvaziamento da pilha.

Pilhas - Operações

- **Criação**;
- Mostrar valores da pilha;
- Inserção de elementos na pilha;
- Remoção de elementos na pilha;
- Descoberta do elemento do topo;
- Esvaziamento da pilha.

Pilha - Criação

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

pilha = Pilha()
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
```

Pilha - Criação

pilha.py

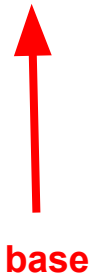
```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
      "elemento 2","elemento 3"]

pilha = Pilha()
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
```

Pilha - Criação

```
self.dados = ["elemento 1", "elemento 2", "elemento 3"]
```



base

Pilha - Criação

```
self.dados = ["elemento 1", "elemento 2", "elemento 3"]
```

Pilha - Criação

```
self.dados = ["elemento 1","elemento 2", "elemento 3"]
```

elemento 3
elemento 2
elemento 1

Exercício 1

Crie uma pilha com os seguintes valores: Amarelo, Vermelho, Verde e Azul.

O valor Amarelo deve estar na base da pilha e Azul no topo.

Pilhas - Operações

- Criação;
- **Mostrar valores da pilha;**
- Inserção de elementos na pilha;
- Remoção de elementos na pilha;
- Descoberta do elemento do topo;
- Esvaziamento da pilha.

Mostrar valores da pilha

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

    def getPilha(self):
        return self.dados

pilha = Pilha()
pilha.getPilha()
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.getPilha()
```

Exercício 2

Mostre os valores da pilha do exemplo anterior (Amarelo, Vermelho, Verde e Azul).

Pilhas - Operações

- Criação;
- Mostrar valores da pilha;
- **Inserção de elementos na pilha**;
- Remoção de elementos na pilha;
- Descoberta do elemento do topo;
- Esvaziamento da pilha.

Atenção!

Inserção de elementos na pilha é sempre feita pelo topo!

Inserção de elementos na pilha

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

    def inserirDados(self, novoElem):
        self.dados.append(novoElem)

pilha = Pilha()
pilha.inserirDados("elemento 4")
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.inserirDados("elemento 4")
```

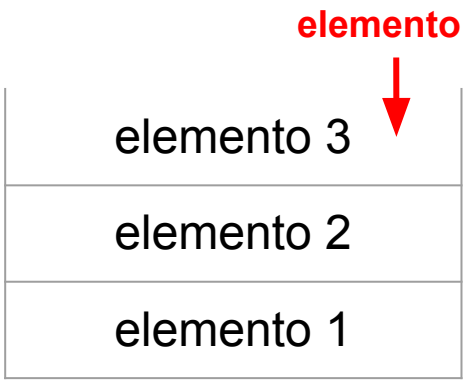
Inserção de elementos na pilha

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

    def inserirDados(self, novoElem):
        self.dados.append(novoElem)

pilha = Pilha()
pilha.inserirDados("elemento 4")
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.inserirDados("elemento 4")
```



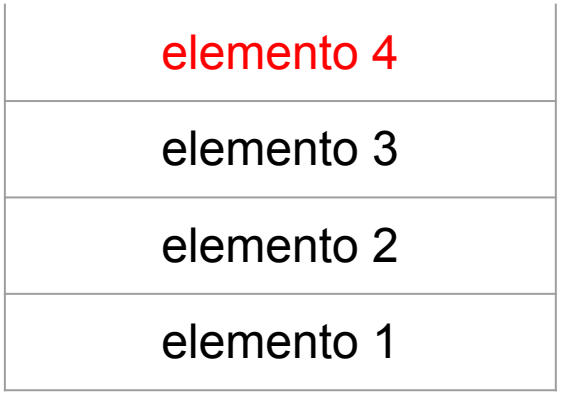
Inserção de elementos na pilha

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

    def inserirDados(self, novoElem):
        self.dados.append(novoElem)

pilha = Pilha()
pilha.inserirDados("elemento 4")
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.inserirDados("elemento 4")
```



Exercício 3

Escreva um programa solicite ao usuário uma sequência de caracteres (você pode definir a quantidade de caracteres). Cada caracter ocupará uma posição da pilha chamada de “letras”.

Pilhas - Operações

- Criação;
- Mostrar valores da pilha;
- Inserção de elementos na pilha;
- **Remoção de elementos na pilha;**
- Descoberta do elemento do topo;
- Esvaziamento da pilha.

Remoção de elementos na pilha

A remoção de elementos na pilha é sempre feita pelo topo!

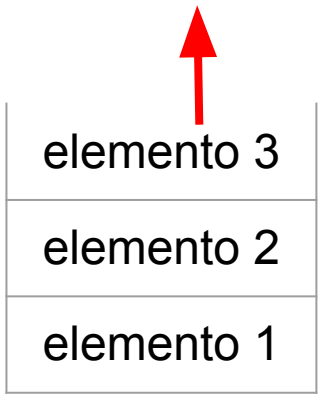
Remoção de elementos na pilha

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

    def removerDados(self):
        self.dados.pop()

pilha = Pilha()
pilha.removerDados()
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.removerDados()
```



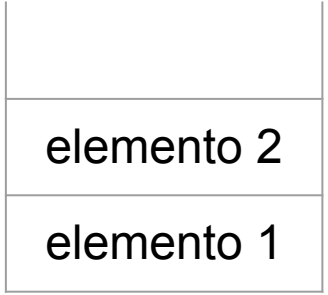
Remoção de elementos na pilha

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1",
     "elemento 2","elemento 3"]

    def removerDados(self):
        self.dados.pop()

pilha = Pilha()
pilha.removerDados()
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.removerDados()
```



Exercício 4

Crie uma pilha com seu nome e seus sobrenomes de tal forma que seu nome fique na base da pilha. Em seguida, remova seu último sobrenome da pilha , que está no topo.

Pilhas - Operações

- Criação;
- Mostrar valores da pilha;
- Inserção de elementos na pilha;
- Remoção de elementos na pilha;
- **Descoberta do elemento do topo;**
- Esvaziamento da pilha.

Descoberta de elementos do topo

pilha.py

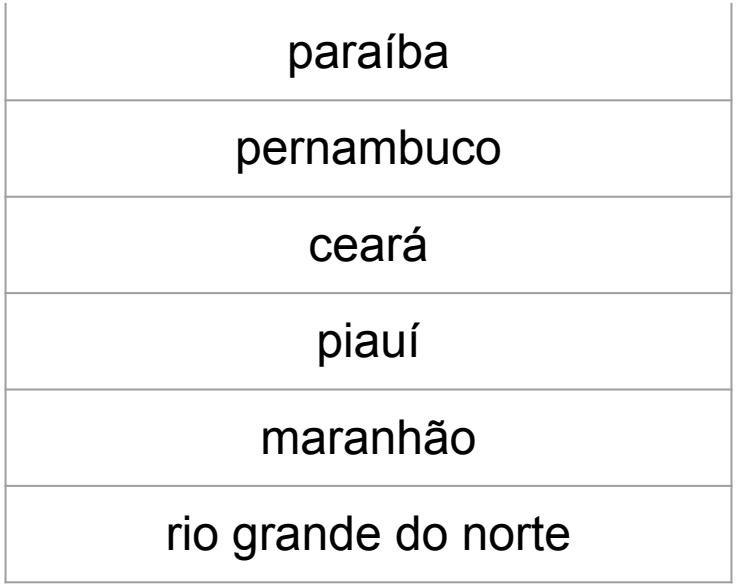
```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1", "elemento
     2","elemento 3"]

    def topo(self):
        return (self.dados[len(self.dados)-1])

pilha = Pilha()
print(pilha.topo())
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
print(pilhaTeste.topo())
```

Exercício 5

Imprima na tela o topo da pilha que contém os seguintes valores:



Pilhas - Operações

- Criação;
- Mostrar valores da pilha;
- Inserção de elementos na pilha;
- Remoção de elementos na pilha;
- Descoberta do elemento do topo;
- **Esvaziamento da pilha.**

Esvaziamento da lista

pilha.py

```
class Pilha(object):
    def __init__(self):
        self.dados = ["elemento 1", "elemento
     2","elemento 3"]

    def esvaziar(self):
        while(len(self.dados) != 0):
            self.dados =  self.dados.pop()

pilha = Pilha()
pilha.esvaziar()
```

programa.py

```
import pilha

pilhaTeste =  pilha.Pilha()
pilhaTeste.esvaziar()
```

Exercício 6

Esvazie a seguinte pilha:

Exercício 7

Desenvolva um algoritmo que informe o tamanho de uma pilha de tamanho dinâmico (o usuário irá informar os valores da pilha)

Exercício 8

Desenvolva um algoritmo para testar se duas pilhas P1 e P2 são iguais. Duas pilhas são iguais se possuem os mesmos elementos, na mesma ordem.

Atividade 1 - Pilha

1. Desenvolva um algoritmo para testar se uma pilha P1 tem mais elementos que uma pilha P2. Considere que P1 e P2 já existem.
O protótipo da função deve ser:

```
def testaMaisElementos(pilha1, pilha2):
# A função retornará 1 para verdadeiro (P1 > P2) e 0 para falso.
```

Atividade 2 - Pilha

2. Desenvolva uma operação para transferir elementos de uma pilha P1 para uma pilha P2 (cópia). Siga o protótipo abaixo:

```
def transferirElementos(pilha1, pilha2, flagErro):
#A função retornará 1 em flagErro para sucesso e 0 para erro
```

Atividade 3 - Pilha

3. Desenvolva um algoritmo para inverter a posição dos elementos de uma pilha P. Você pode criar pilhas auxiliares, se necessário. Mas o resultado precisa ser dado na pilha P.

```
def inverter (pilha):
```

Dúvidas?

alanamm.prof@gmail.com