Algoritmos e Estrutura de Dados II

Aula 06

Pilhas

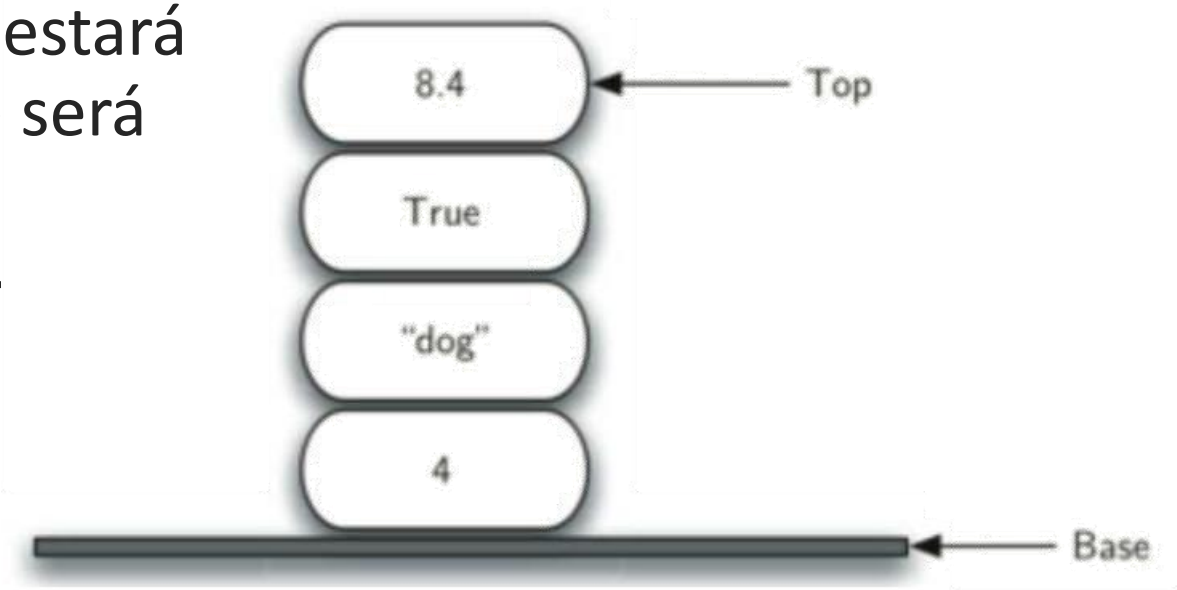
Prof. Dr. Dilermando Piva Jr

2º Semestre - CDN

Várias pilhas de livros...

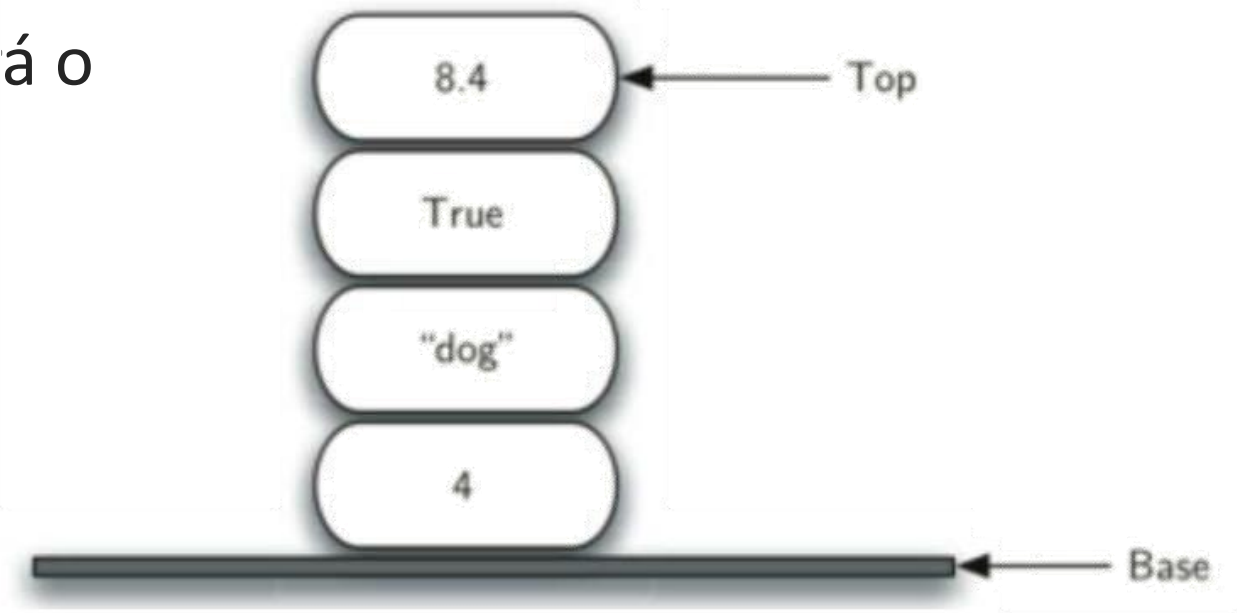
Pilhas

- Uma pilha (stack) e uma estrutura de dados linear em que a inserção e a remoção de elementos e realizada sempre na mesma extremidade, comumente de chamada de **topo**. O oposto do topo e a **base**.
- Quanto mais próximo da base está um elemento, há mais tempo ele está armazenado na estrutura.
- Por outro lado, um item inserido agora estará sempre no topo, o que significa que ele será o primeiro a ser removido
  (se não empilharmos novos elementos antes).



Pilhas

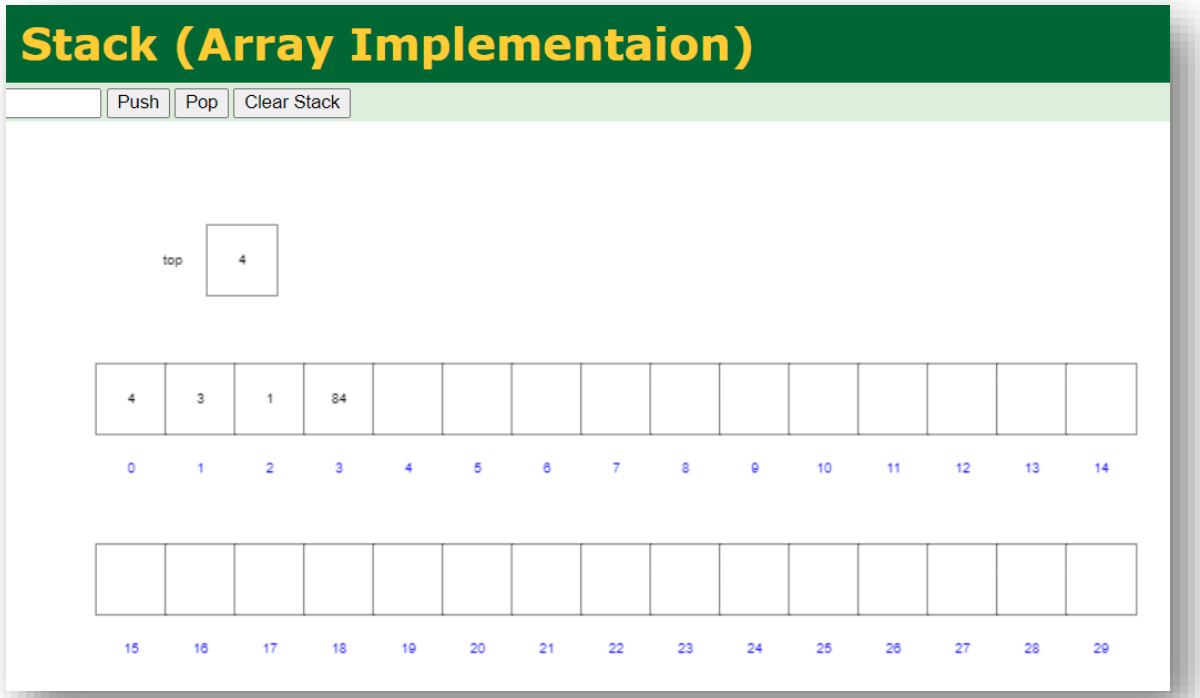
- Por esse princípio de ordenação inerente das pilhas, elas são conhecidas como a estrutura **LIFO**, do inglês, *Last In First Out*, ou seja, o último a entrar é o primeiro a sair... Também pode ser visto como: “primeiro a entrar e último a sair” (**FILO – First in Last out**).
- Essa intuição faz total sentido com uma pilha de livros (como visto no início da aula). O primeiro livro a ser empilhado fica na base da pilha e será o último a ser retirado.
- A ordem de remoção e o inverso da ordem de inserção.

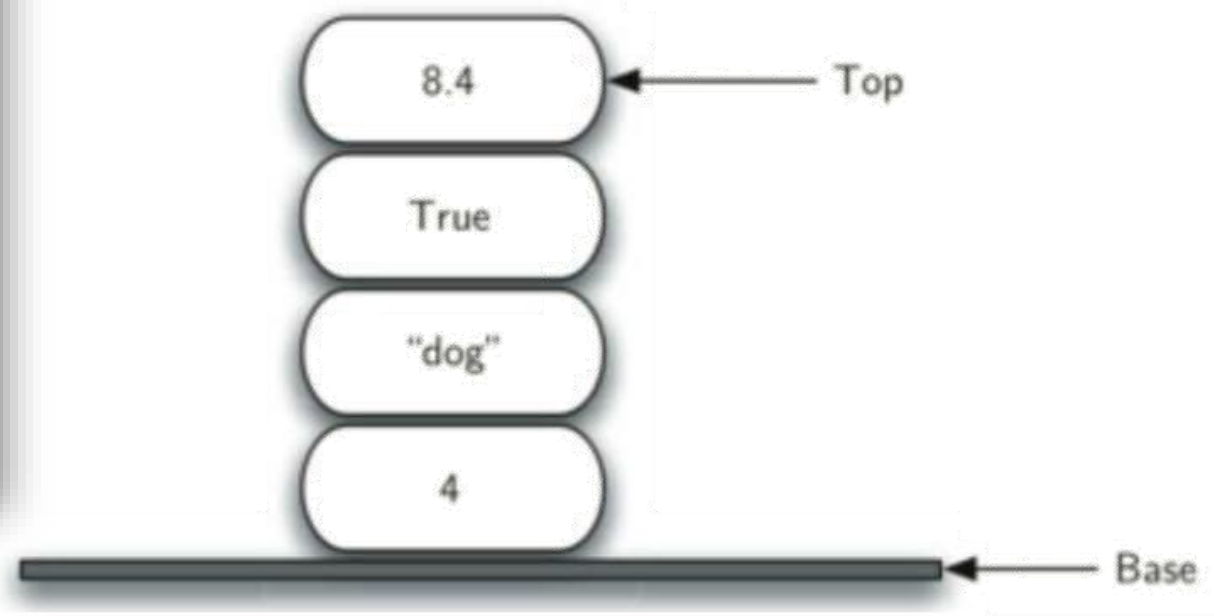


Pilhas

- Link para demonstração...

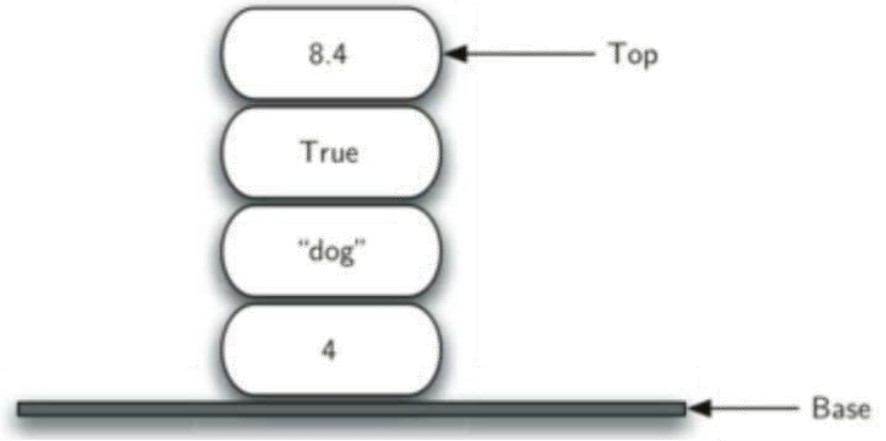
https://www.cs.usfca.edu/~galles/visualization/StackArray.html





Pilhas

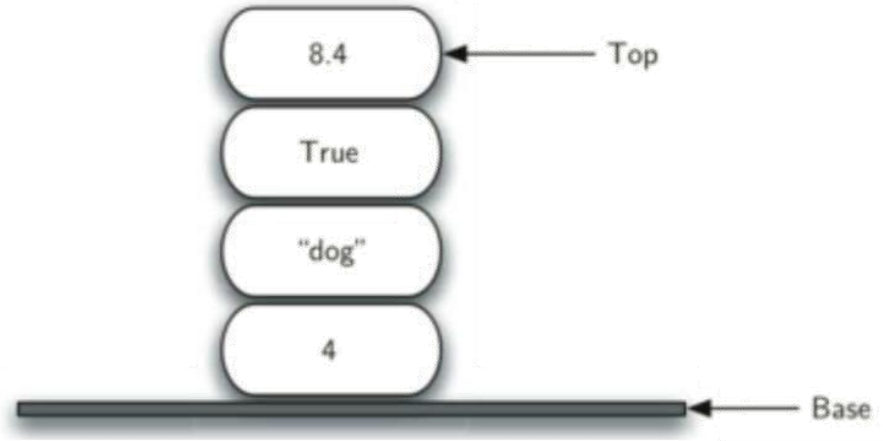
- Para que possamos implementar um classe Pilha (**Stack**), devemos ter em mente quais suas variáveis internas e quais as operações que podemos aplicar sobre os seus elementos.
- A listagem a seguir mostra como podemos construir a pilha da figura ao lado, a partir de uma pilha inicialmente vazia.
- Note que podemos utilizar uma lista **S** para armazenar os elementos da pilha.



Operação	Conteúdo	Retorno	Descrição
s.is_empty()	[]	True	Verifica se pilha está vazia
s.push(4)	[4]		Insere elemento no topo
s.push('dog')	[4, 'dog']		Insere elemento no topo
s.peek()	[4, 'dog']	'dog'	Consulta o elemento do topo (mantém)
s.push(True)	[4, 'dog', True]		Insere elemento no topo
s.size()	[4, 'dog', True]	3	Retorna número de elementos da pilha
s.is_empty()	[4, 'dog', True] False		Verifica se pilha está vazia
s.puch(8.4)	[4, 'dog', True, 8.4]		Insere elemento no topo
s.pop()	[4, 'dog', True]	8.4	Remove elemento do topo
s.pop()	[4, 'dog']	True	Remove elemento do topo
s.size()	[4, 'dog']	2	Retorna número de elementos da pilha

Pilhas

- Para que possamos implementar um classe Pilha (**Stack**), devemos ter em mente quais suas variáveis internas e quais as operações que podemos aplicar sobre os seus elementos.
- A listagem a seguir mostra como podemos construir a pilha da figura ao lado, a partir de uma pilha inicialmente vazia.
- Note que podemos utilizar uma lista **S** para armazenar os elementos da pilha.



Operação	Conteúdo	Retorno	Descrição
s.is_empty()	[]	True	Verifica se pilha está vazia
s.push(4)	[4]		Insere elemento no topo
s.push('dog')	[4, 'dog']		Insere elemento no topo
s.peek()	[4, 'dog']	'dog'	Consulta o elemento do topo (mantém)
s.push(True)	[4, 'dog', True]		Insere elemento no topo
s.size()	[4, 'dog', True]	3	Retorna número de elementos da pilha
s.is_empty()	[4, 'dog', True] False		Verifica se pilha está vazia
s.puch(8.4)	[4, 'dog', True, 8.4]		Insere elemento no topo
s.pop()	[4, 'dog', True]	8.4	Remove elemento do topo
s.pop()	[4, 'dog']	True	Remove elemento do topo
s.size()	[4, 'dog']	2	Retorna número de elementos da pilha

Pilhas

- Para implementar uma pilha em Python, iremos utilizar como atributo uma lista chamada `items`, que inicia vazia.
- Definiremos os métodos `push()` e `pop()` para inserção e remoção de elementos do topo, bem como `peek()`, `size()` e `is_empty()`, para consultar o elemento do topo, obter o número de elementos da pilha e verifica se a pilha está vazia.
- Note que na nossa implementação de pilha, tanto a inserção quanto a remoção tem complexidade O(1).

Pilhas

```
# Implementacao da classe Pilha
class Stack:
# Inicia com uma pilha vazia
def __init__(self):
    self.itens = []


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


# Adiciona elemento no topo (topo e o final da lista)
def push(self, item):
    self.itens.append(item)
    print('PUSH %s' %item)
```

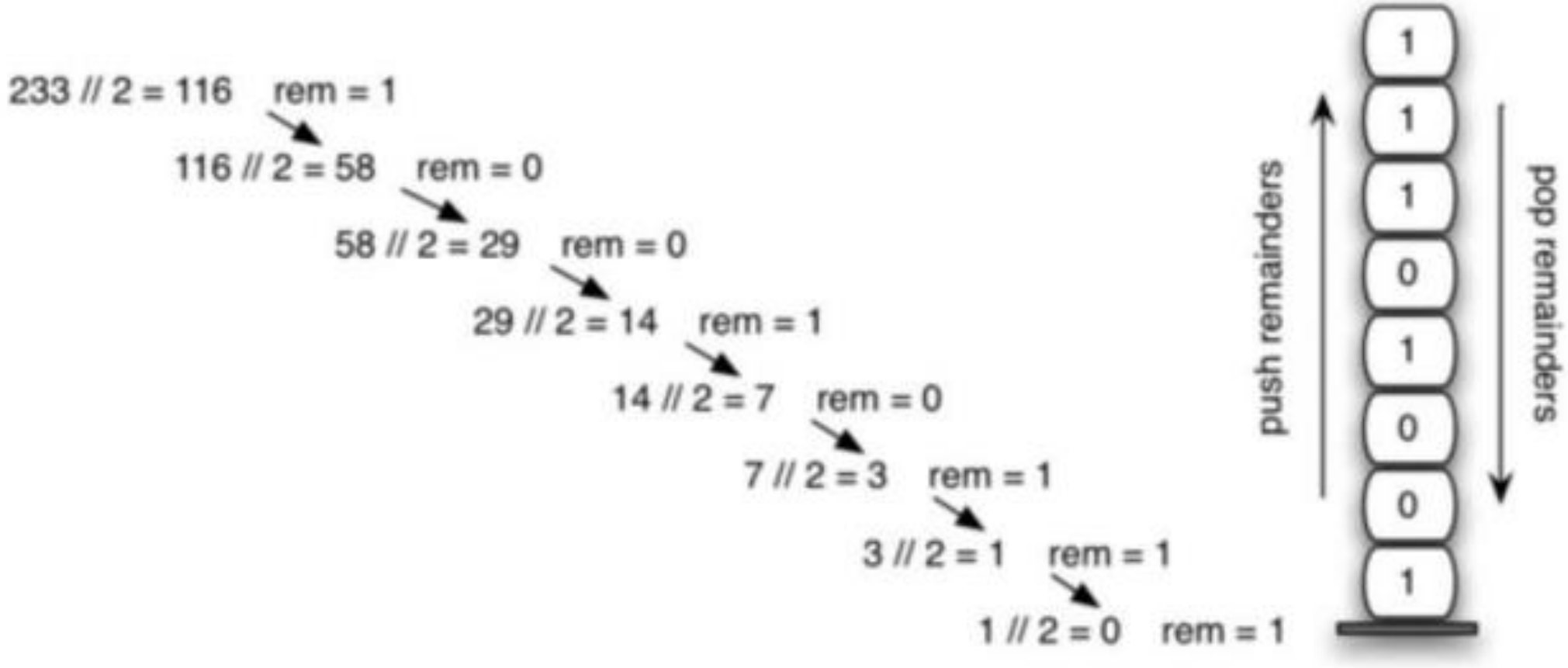
```
# Remove elemento do topo (final da lista)
def pop(self):
    print('POP')
    return self.itens.pop()


# Obtem o elemento do topo (mas nao remove)
def peek(self):
# Em Python, indice -1 retorna ultimo elemento (topo)
    return self.itens[-1]


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


# Imprime pilha na tela
def print_stack(self):
    print(self.itens)
```

```
# TESTANDO
S = Stack()
S.print_stack()
S.push(1)
S.push(2)
S.push(3)
S.print_stack()
S.pop()
S.pop()
S.print_stack()
S.push(7)
S.push(8)
S.push(9)
S.print_stack()
print(S.is_empty())
```

VAMOS PARA A PRÁTICA ?!!!

Aplicações de Pilhas
Conversão de decimal para binário

Converter 233 em Binário = 11101001



Aplicações de Pilhas

Conversão de decimal para binário

```
from Pilha import Stack

# Função que converte um número decimal para binário
def decimal_binario(numero):
 s = Stack()
 while numero > 0:
    resto = numero % 2
    s.push(resto)
    numero = numero // 2
 binario = ''
 while not s.is_empty():
     binario = binario + str(s.pop())
 return binario
```

```
# TESTANDO
n = int(input('Entre com um número inteiro: ‘))
print(decimal_binario(n))
```