

# Stack

William Fiset

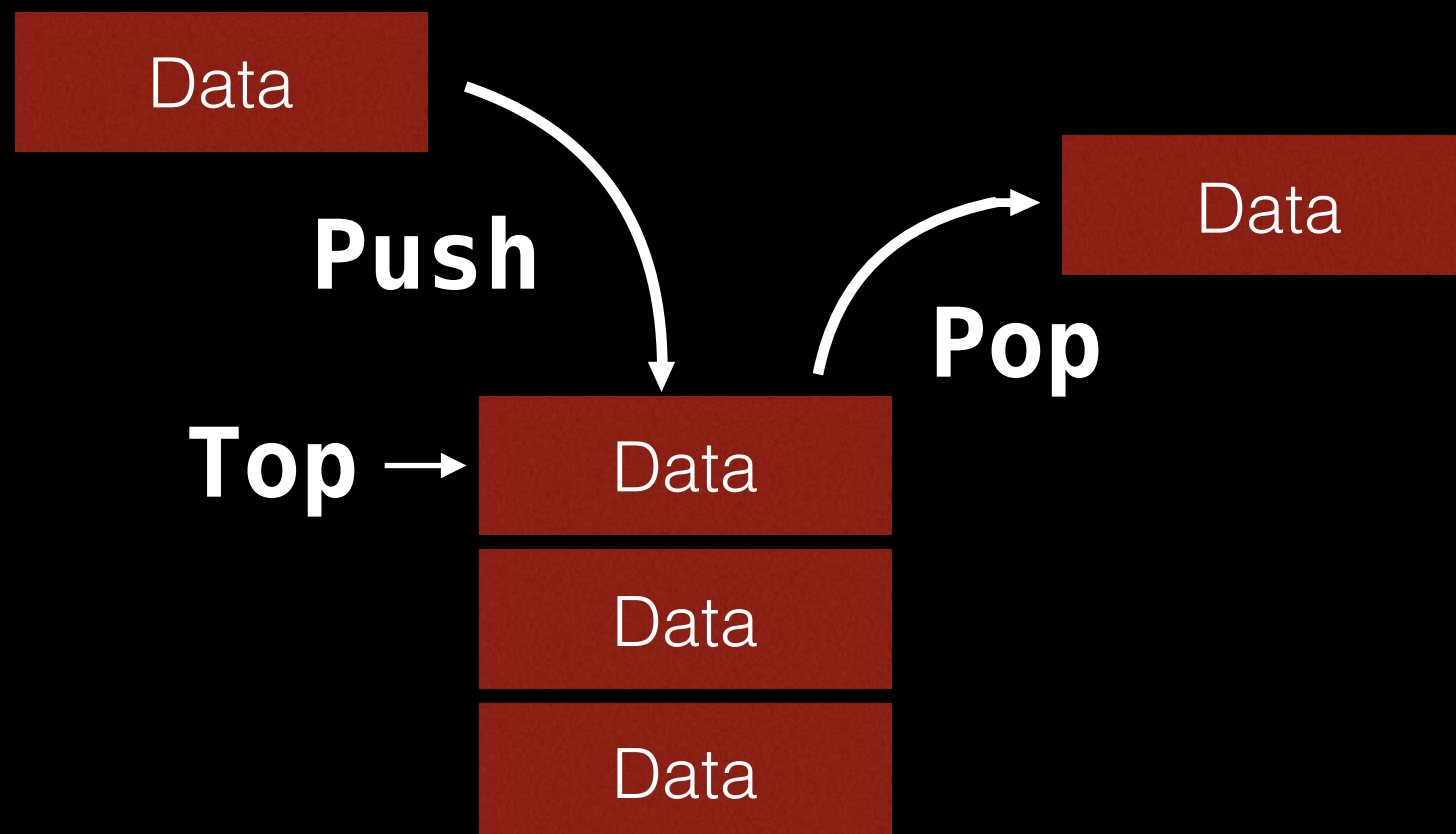
# Outline

- Discussion about Stacks
  - What is a Stack?
  - When and where is a Stack used?
  - Complexity Analysis
  - Stack usage examples
- Implementation details
  - Pushing elements on stack
  - Popping elements from stack
- Code Implementation

# Discussion

# What is a Stack?

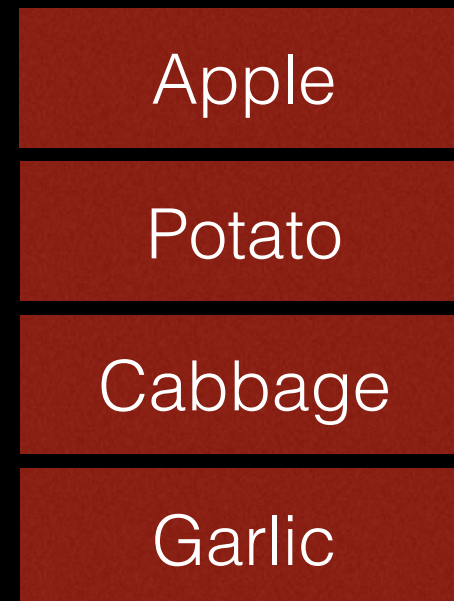
A stack is a one-ended linear data structure which models a real world stack by having two primary operations, namely **push** and **pop**.



# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')
```



# What is a Stack?

## Instructions

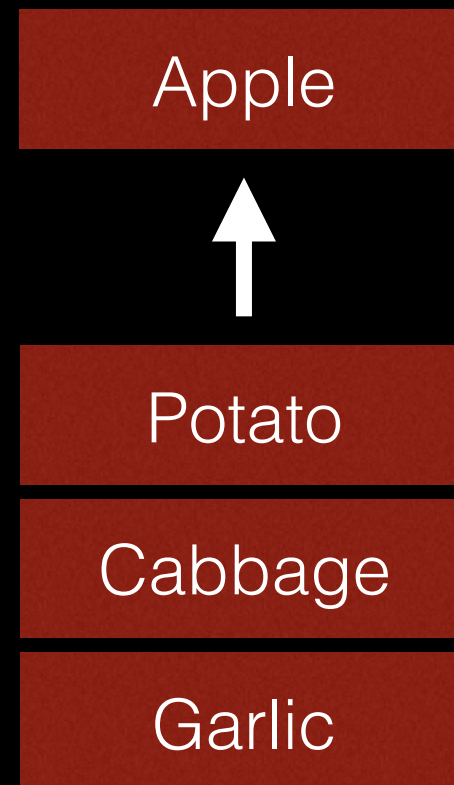
→ pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')

Apple
Potato
Cabbage
Garlic

# What is a Stack?

## Instructions

→ pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')



# What is a Stack?

## Instructions

→ pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')

Potato

Cabbage

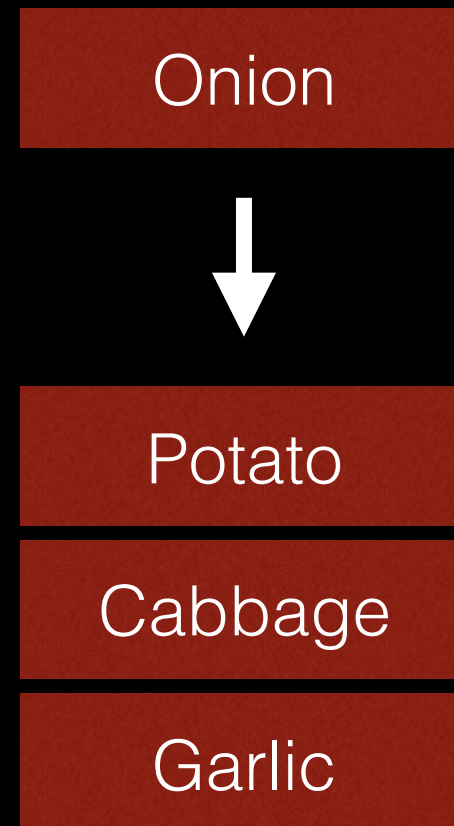
Garlic



# What is a Stack?

## Instructions

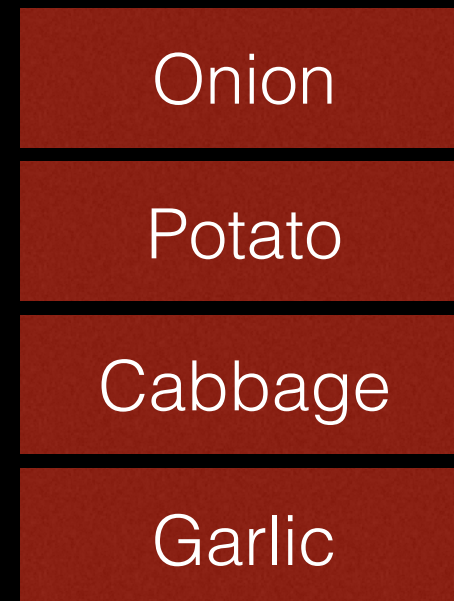
→ pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')



# What is a Stack?

## Instructions

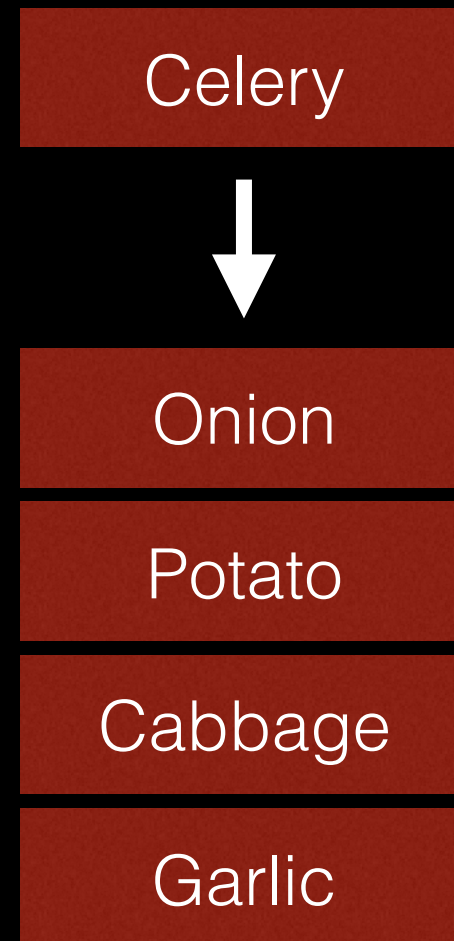
→ pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')



# What is a Stack?

## Instructions

pop()  
push('Onion')  
→ push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')



# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
→ push('Celery')  
push('Watermelon')  
pop()  
pop()  
push('Lettuce')
```

Celery

Onion

Potato

Cabbage

Garlic

# What is a Stack?

## Instructions

pop()  
push('Onion')  
push('Celery')  
→ push('Watermelon')  
pop()  
pop()  
push('Lettuce')

Watermelon



Celery

Onion

Potato

Cabbage

Garlic

# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
→ push('Watermelon')  
pop()  
pop()  
push('Lettuce')
```

Watermelon

Celery

Onion

Potato

Cabbage

Garlic

# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
→ pop()  
pop()  
push('Lettuce')
```

Watermelon

Celery

Onion

Potato

Cabbage

Garlic

# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
→ pop()  
pop()  
push('Lettuce')
```

Watermelon



Celery

Onion

Potato

Cabbage

Garlic



# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
→ pop()  
pop()  
push('Lettuce')
```

Celery

Onion

Potato

Cabbage

Garlic

# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
→ pop()  
push('Lettuce')
```

Celery

Onion

Potato

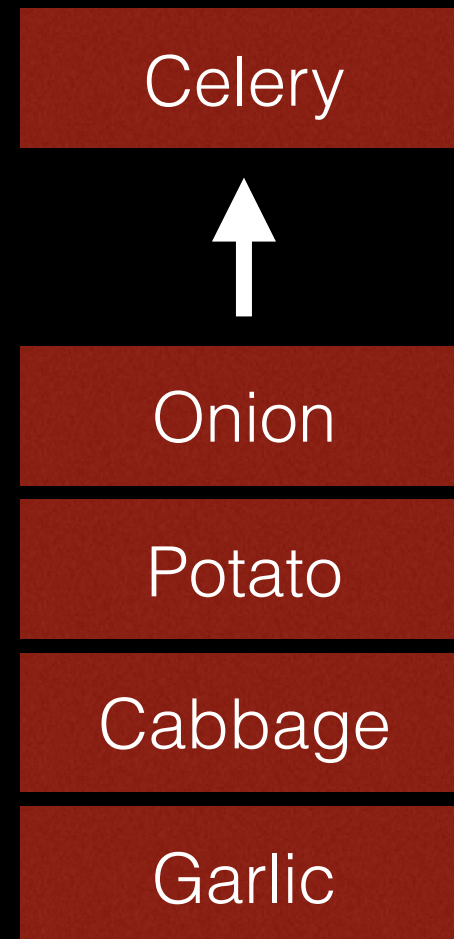
Cabbage

Garlic

# What is a Stack?

## Instructions

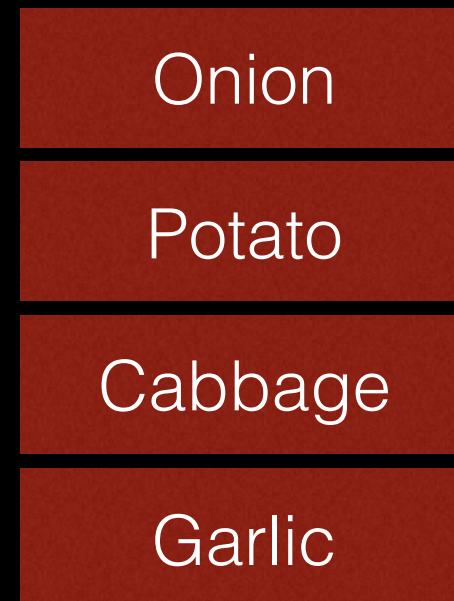
```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
→ pop()  
push('Lettuce')
```



# What is a Stack?

## Instructions

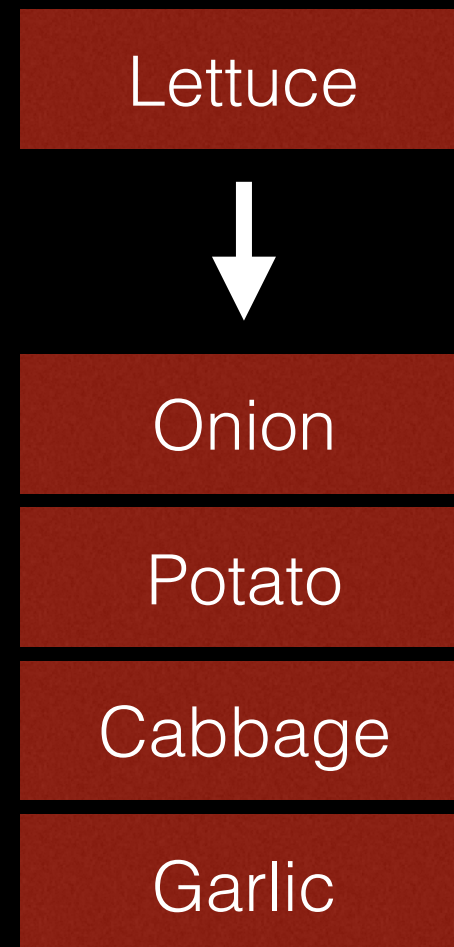
```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
→ pop()  
push('Lettuce')
```



# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
→ push('Lettuce')
```



# What is a Stack?

## Instructions

```
pop()  
push('Onion')  
push('Celery')  
push('Watermelon')  
pop()  
pop()  
→ push('Lettuce')
```

Lettuce

Onion

Potato

Cabbage

Garlic

# When and where is a Stack used?

- Used by undo mechanisms in text editors.
- Used in compiler syntax checking for matching brackets and braces.
- Can be used to model a pile of books or plates.
- Used behind the scenes to support recursion by keeping track of previous function calls.
- Can be used to do a Depth First Search (DFS) on a graph.

# Complexity Analysis



# Complexity

<b>Pushing</b>	$O(1)$
<b>Popping</b>	$O(1)$
<b>Peeking</b>	$O(1)$
<b>Searching</b>	$O(n)$
<b>Size</b>	$O(1)$

# Example – Brackets

**Problem:** Given a string made up of the following brackets: `()[]{}` , determine whether the brackets properly match.

`[{}]`  **Valid**

`((()))`  **Valid**

`{}`  **Invalid**

`[()])()`  **Invalid**

`[]{}({})`  **Valid**

# Example – Brackets

Bracket Sequence:

$[[\{\}]()]$

Current Bracket:  $\emptyset$

Reversed Bracket:  $\emptyset$

# Example – Brackets

Bracket Sequence:

**[** [ { } ] ( ) ]

Current Bracket: [

Reversed Bracket: ]



[

# Example – Brackets

Bracket Sequence:

**[** [ { } ] ( ) ]

Current Bracket: [

Reversed Bracket: ]



# Example – Brackets

Bracket Sequence:

`[[{ }]()`

Current Bracket: {

Reversed Bracket: }

{

[

[

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: }

Reversed Bracket: {

}

[

[

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: }

Reversed Bracket: {



[

[



# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: ]

Reversed Bracket: [



[

[

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: `]`

Reversed Bracket: `[`



`[`

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: (

Reversed Bracket: )



(

[

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: )

Reversed Bracket: (



(

[

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: )

Reversed Bracket: (



[

# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: ]

Reversed Bracket: [



# Example – Brackets

Bracket Sequence:

`[[{}]()]`

Current Bracket: ]

Reversed Bracket: [

# Example – Brackets

Bracket Sequence:

`[[{}]()]`  **Valid**

Current Bracket: ]

Reversed Bracket: [



# Example – Brackets

Bracket Sequence:

[{ }) [ ]

Current Bracket:  $\emptyset$

Reversed Bracket:  $\emptyset$

# Example – Brackets

Bracket Sequence:

[{ }) [ ]

Current Bracket: [

Reversed Bracket: ]



[

# Example – Brackets

Bracket Sequence:

[{ }) [ ]

Current Bracket: {

Reversed Bracket: }



# Example – Brackets

Bracket Sequence:

[{ }) [ ]

Current Bracket: }

Reversed Bracket: {



# Example – Brackets

Bracket Sequence:

[{ }) [ ]

Current Bracket: }

Reversed Bracket: {



[

# Example – Brackets

Bracket Sequence:

[{ }) [ ]

Current Bracket: )

Reversed Bracket: (



[

# Example – Brackets

Bracket Sequence:

[{ }) [ ] → Invalid

Current Bracket: )

Reversed Bracket: (

[

# Example – Brackets

Let  $S$  be a stack

```
For bracket in bracket_string:
```

```
    rev = getReversedBracket(bracket)
```

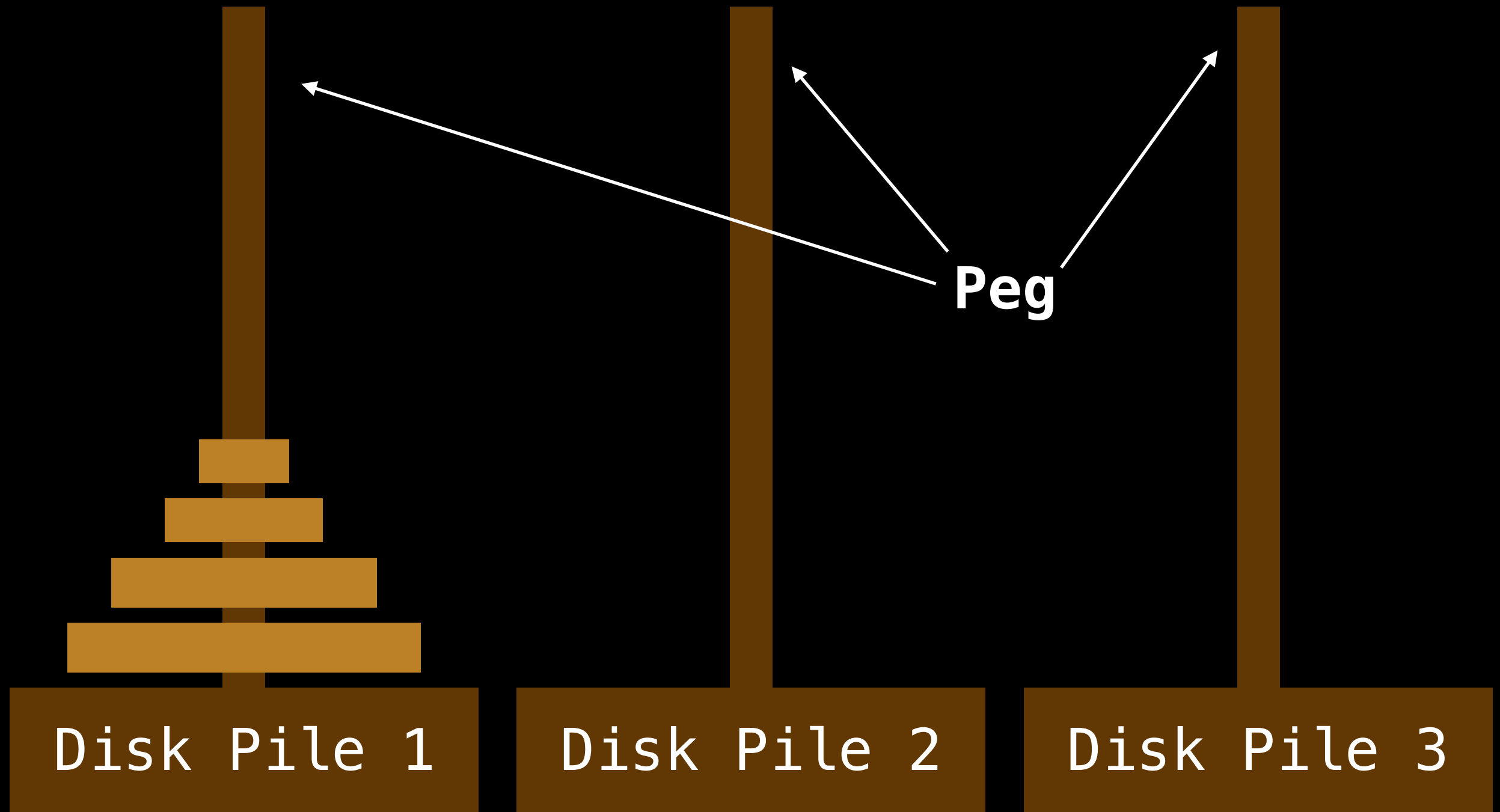
```
    If isLeftBracket(bracket):  
        S.push(bracket)
```

```
    Else If S.isEmpty() or S.pop() != rev:  
        return false // Invalid
```

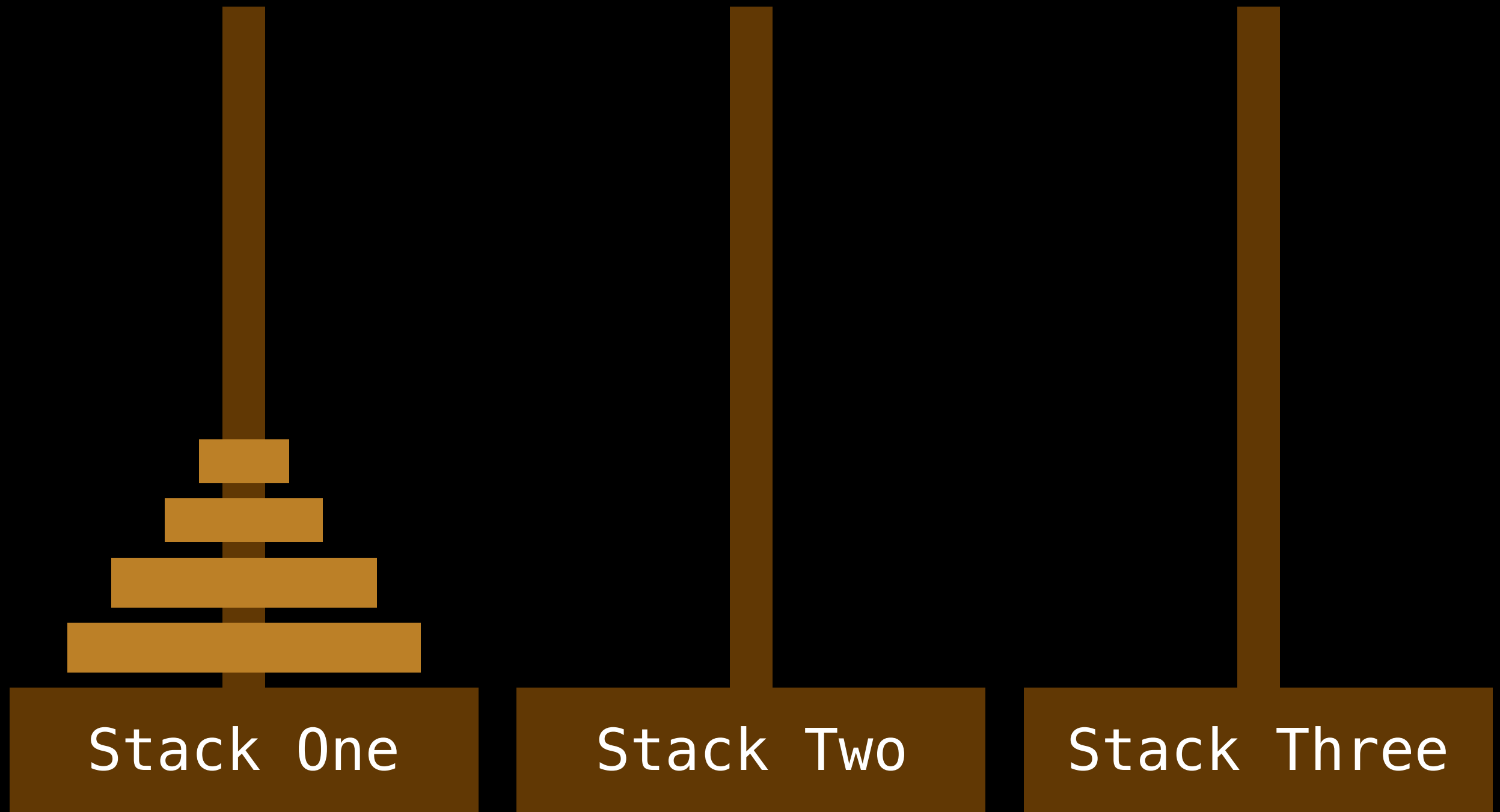
```
return S.isEmpty() // Valid if S is empty
```



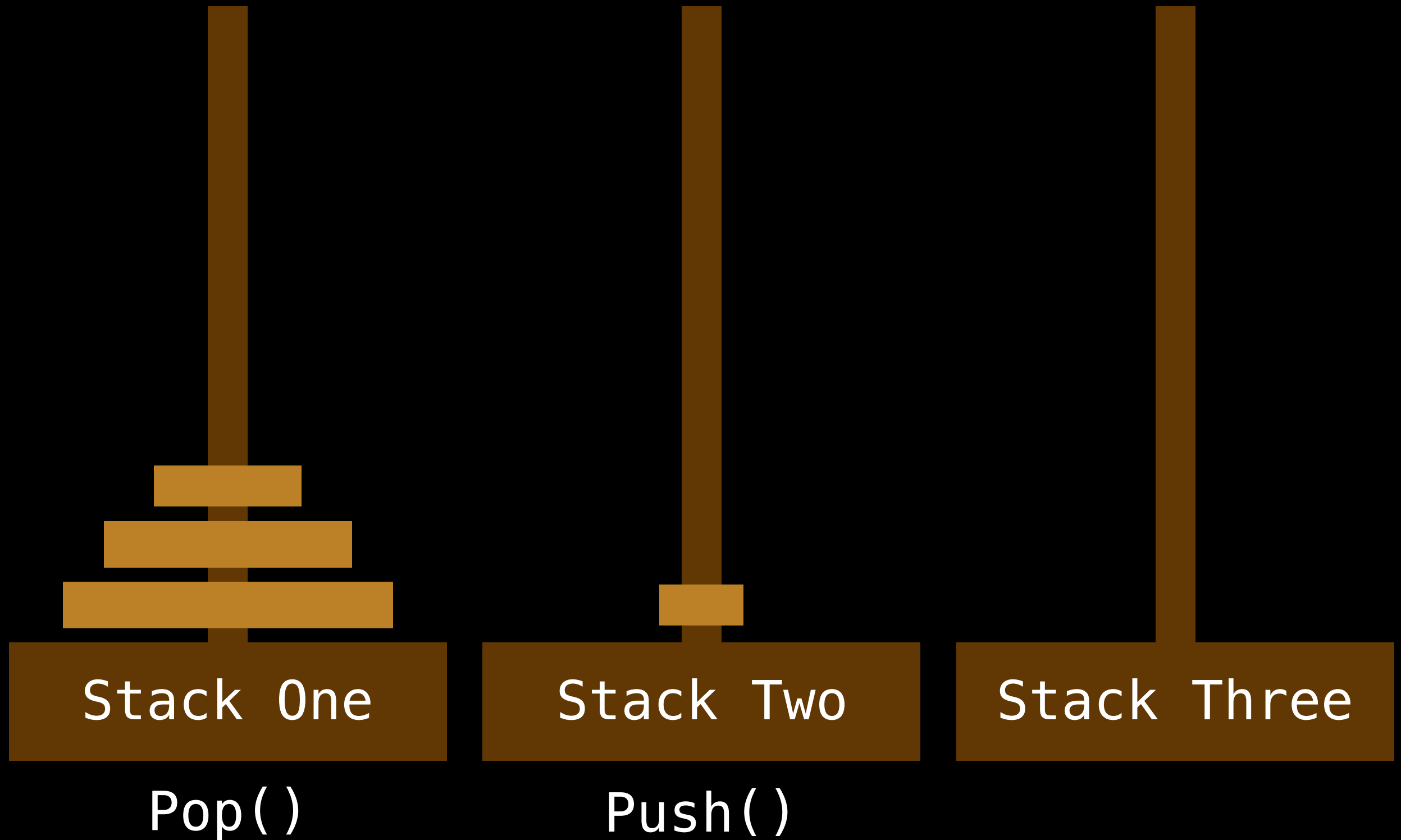
# Tower of Hanoi



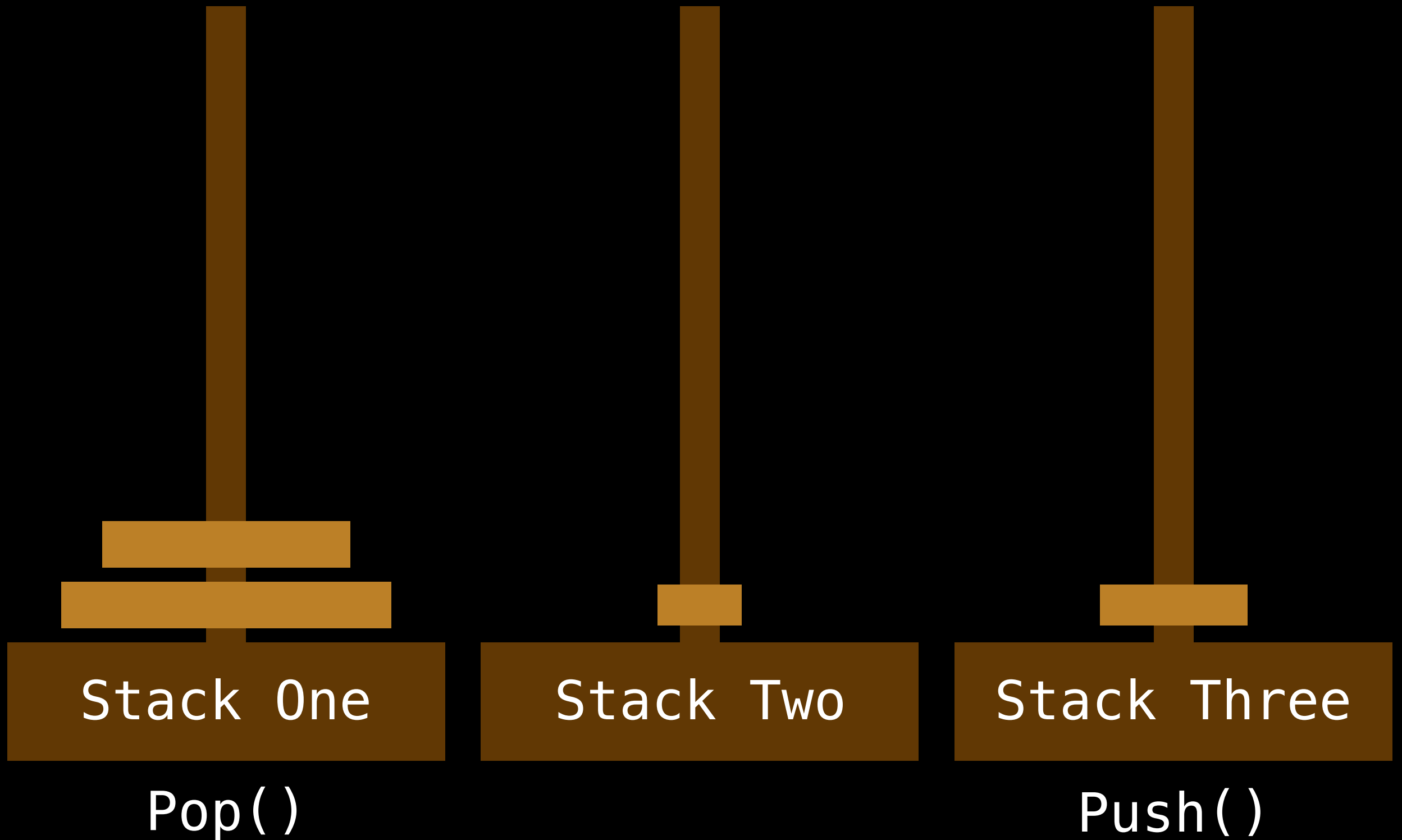
# Tower of Hanoi



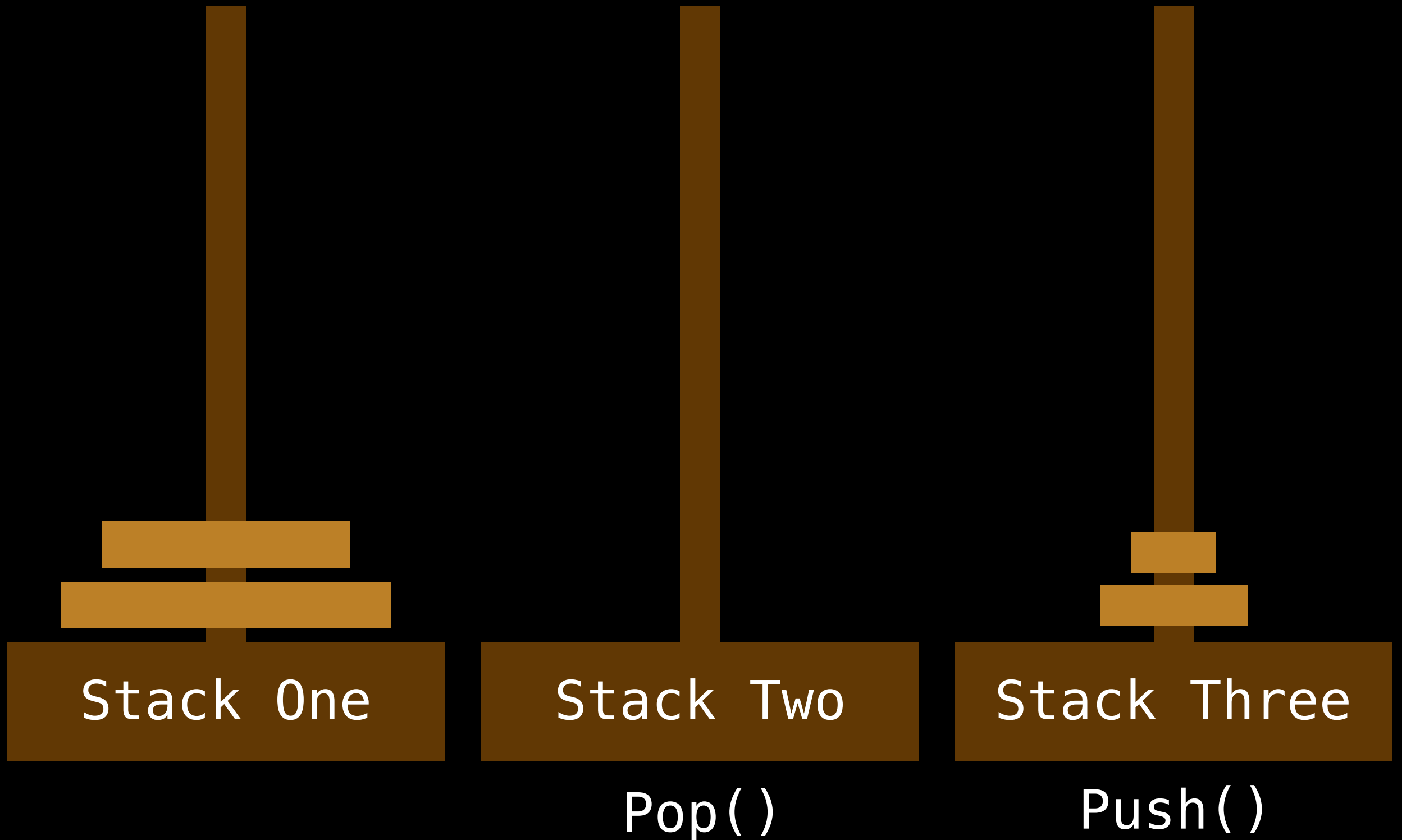
# Tower of Hanoi



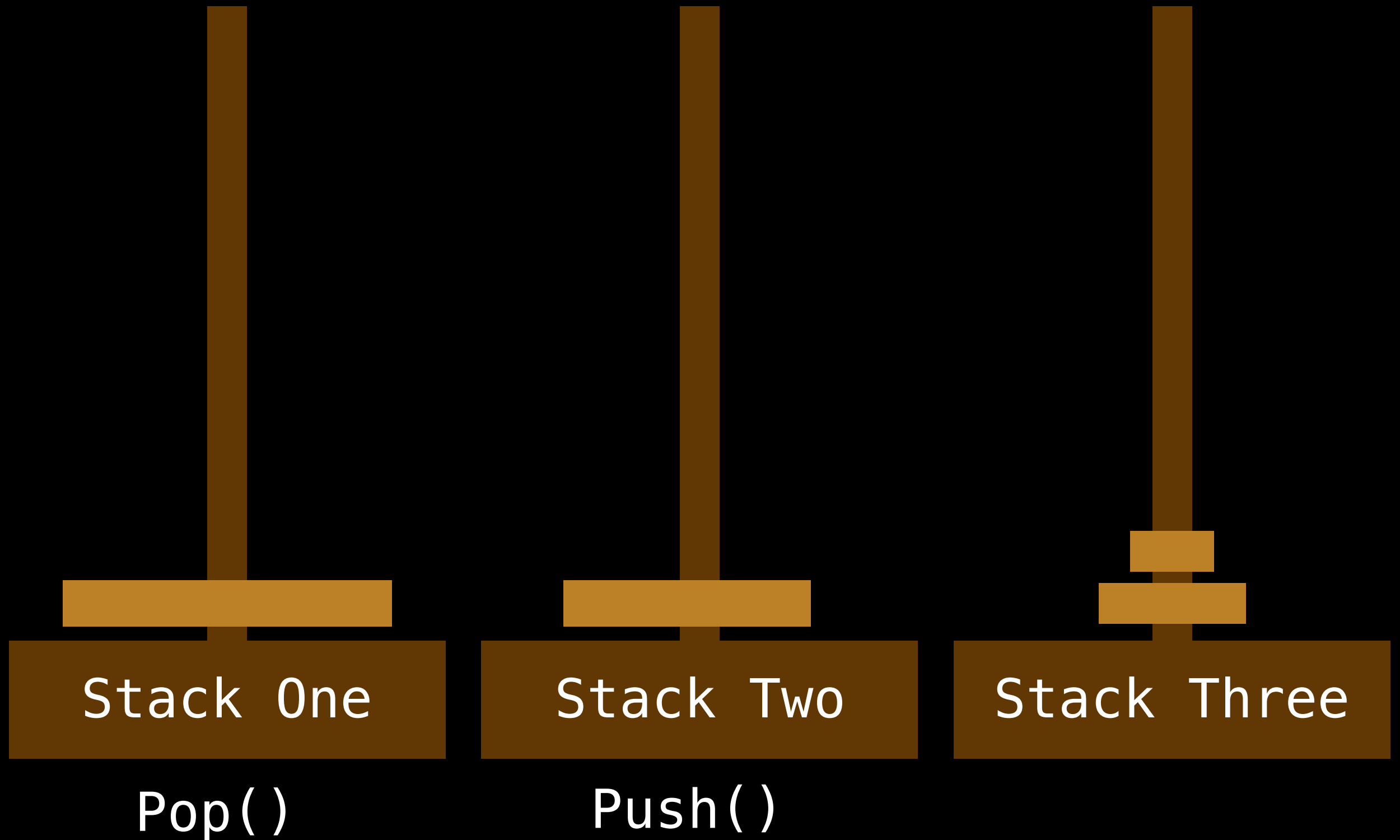
# Tower of Hanoi



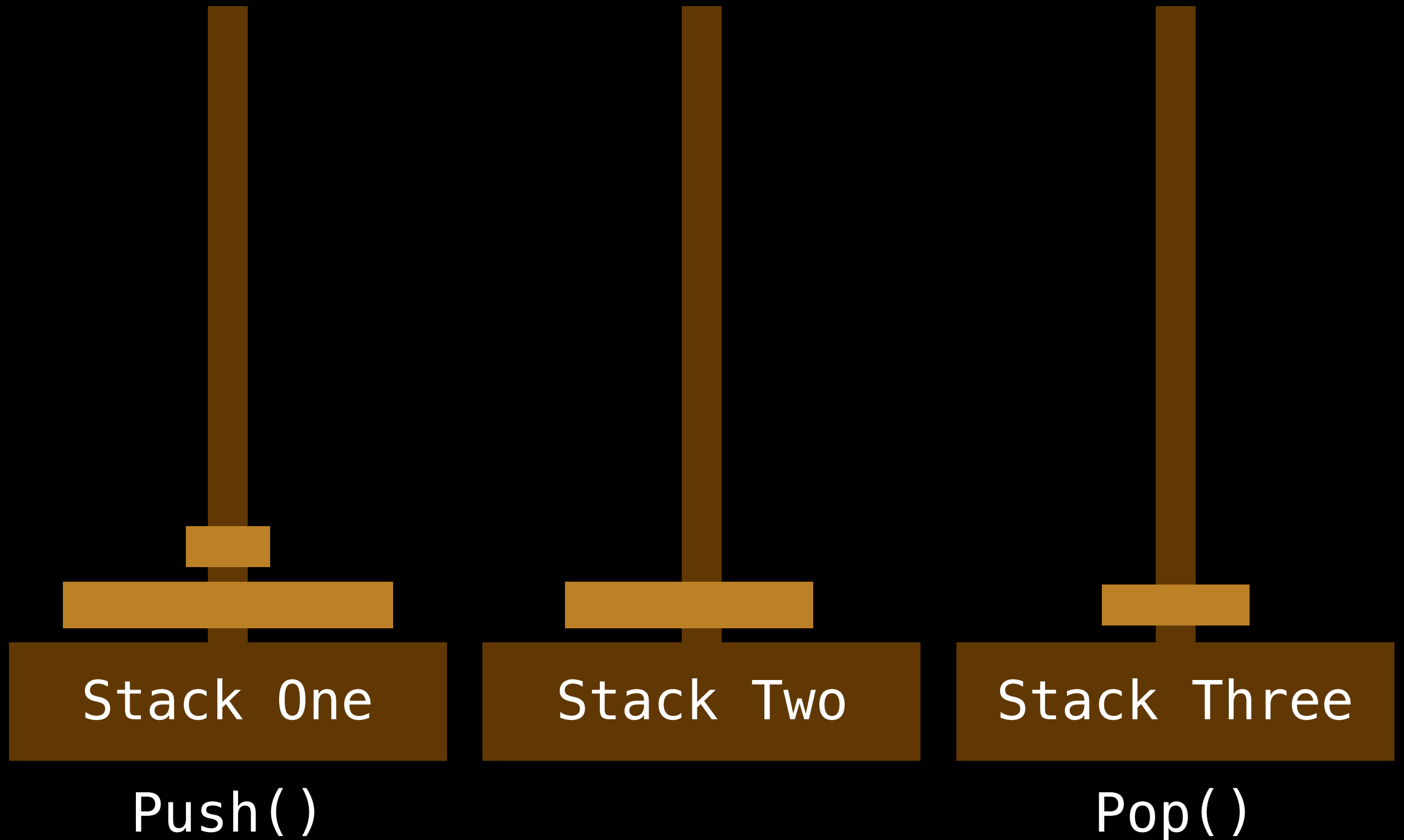
# Tower of Hanoi



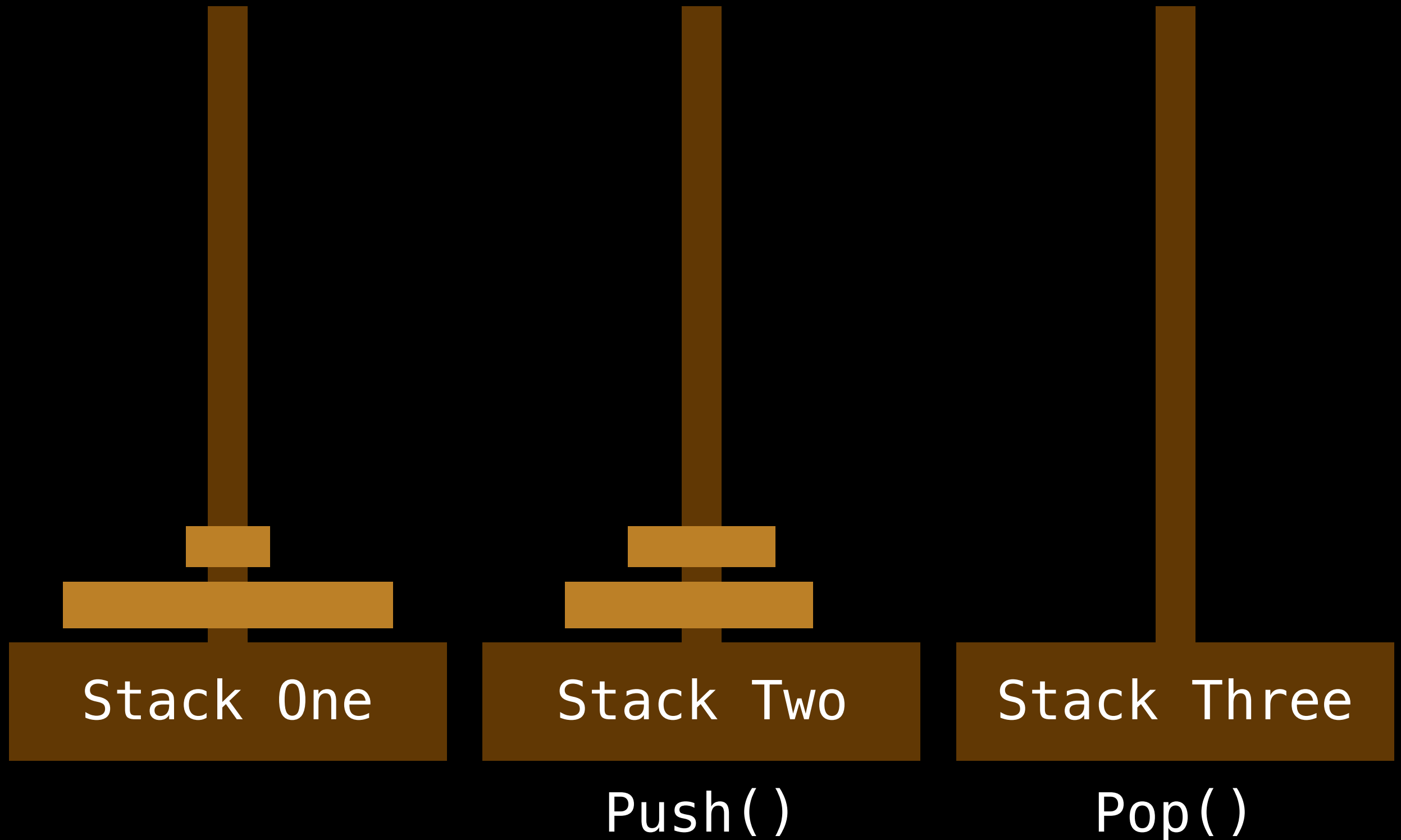
# Tower of Hanoi



# Tower of Hanoi

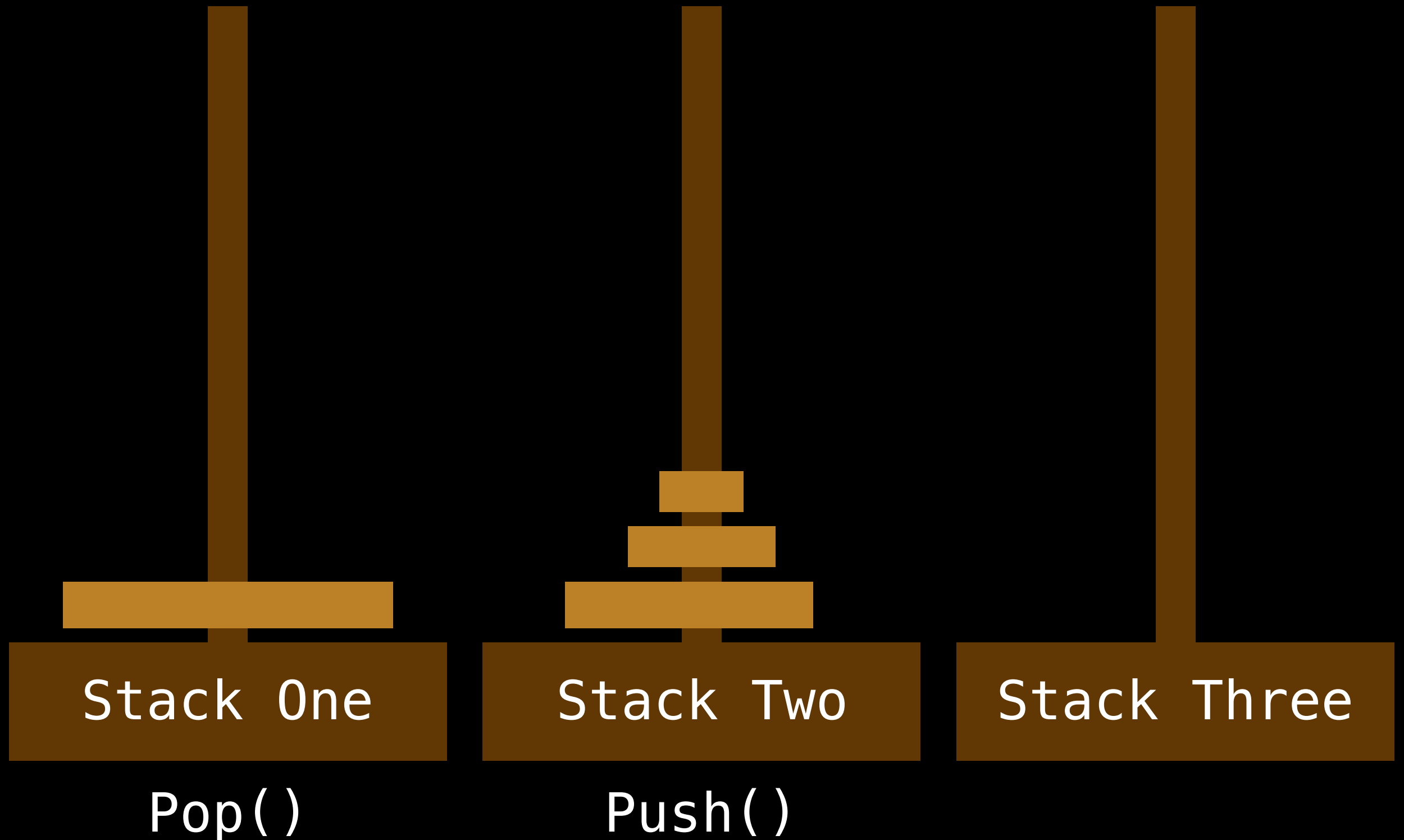


# Tower of Hanoi

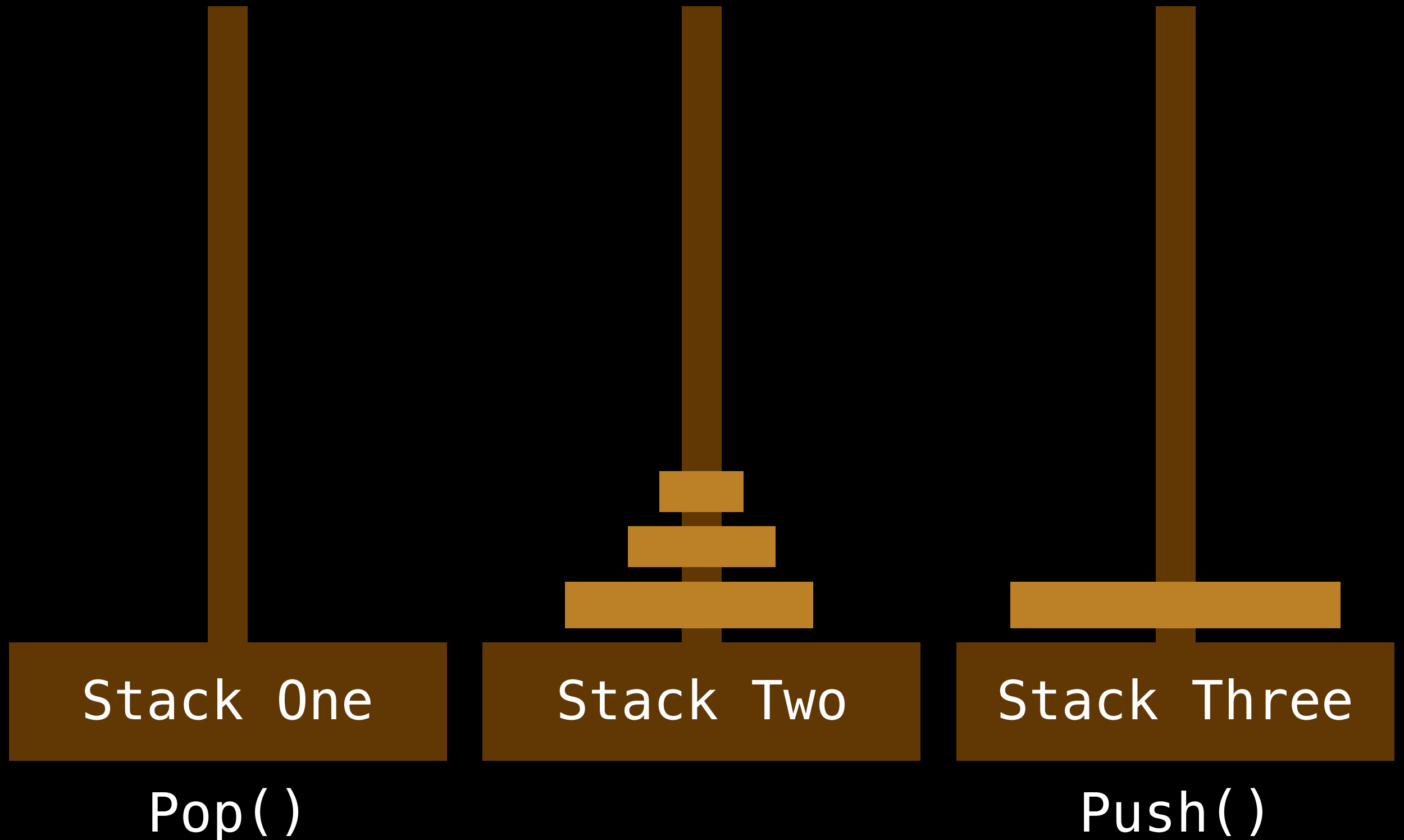




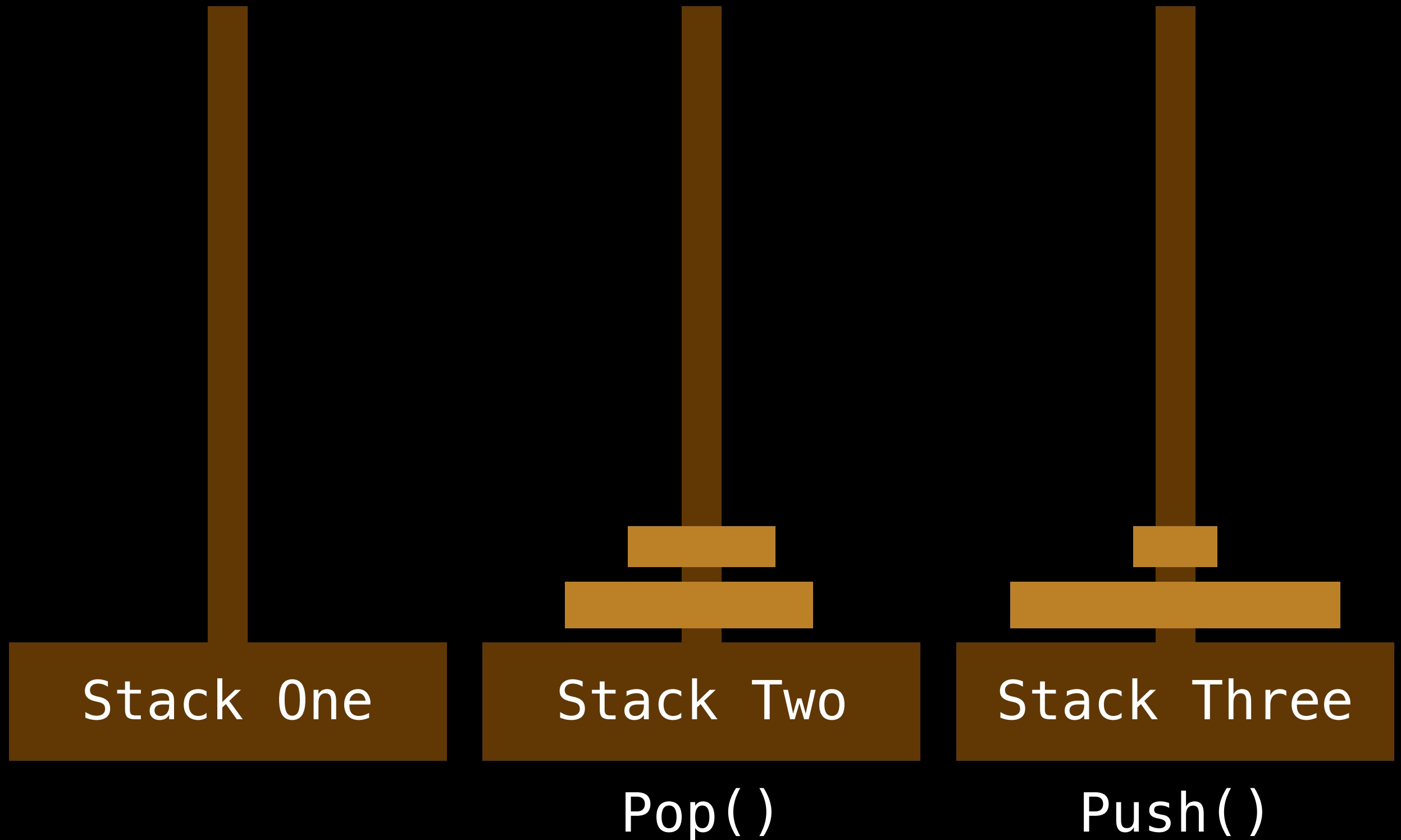
# Tower of Hanoi



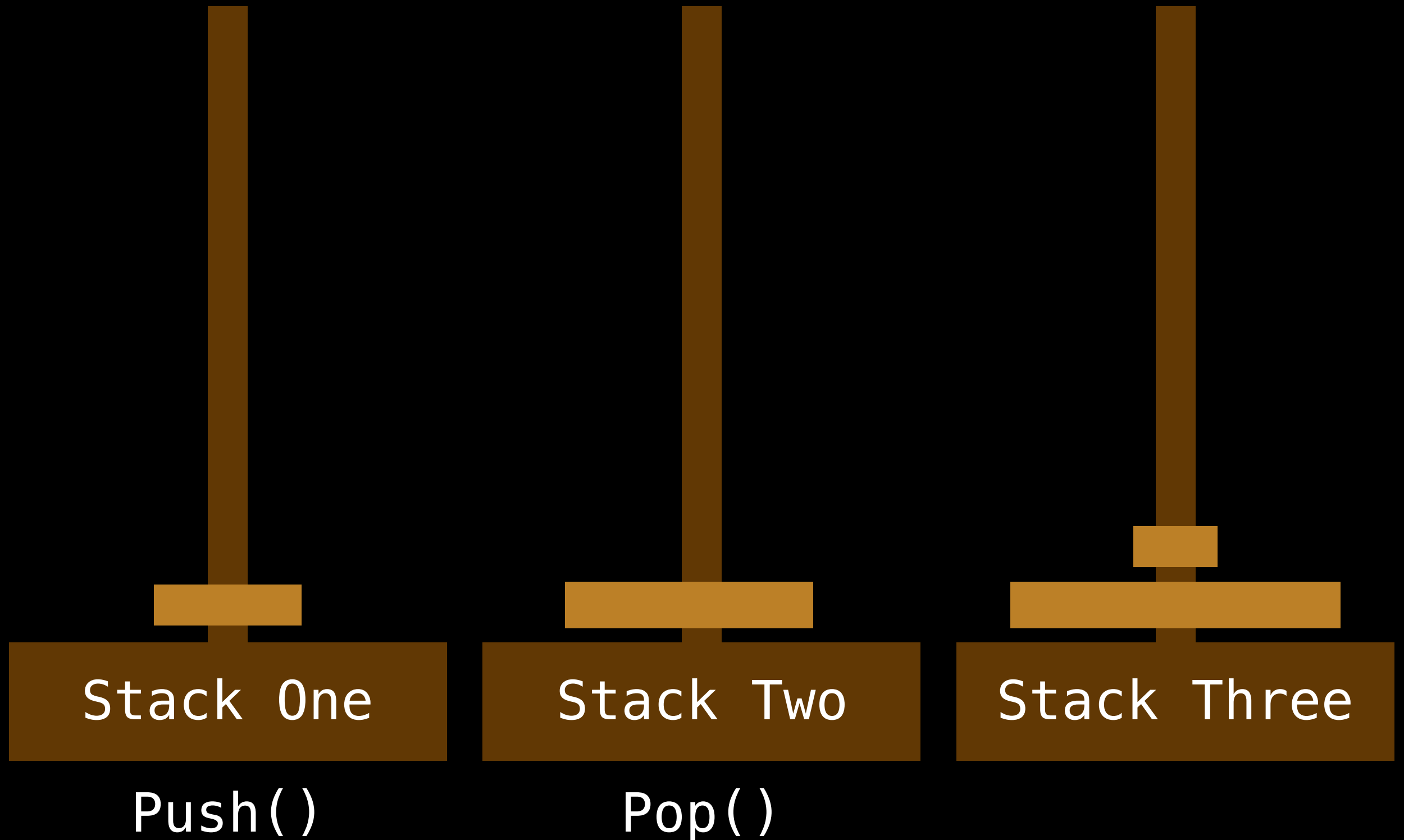
# Tower of Hanoi



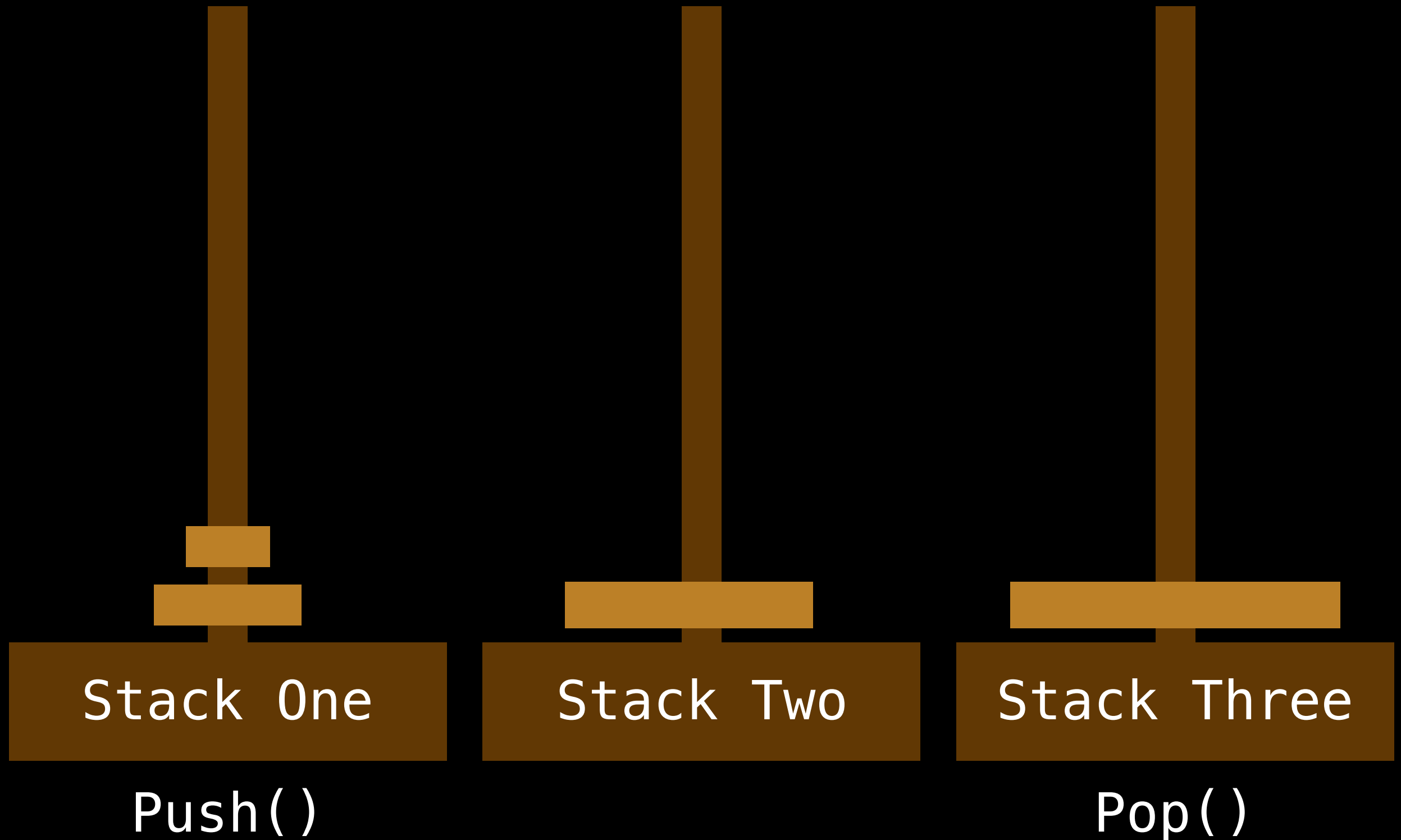
# Tower of Hanoi



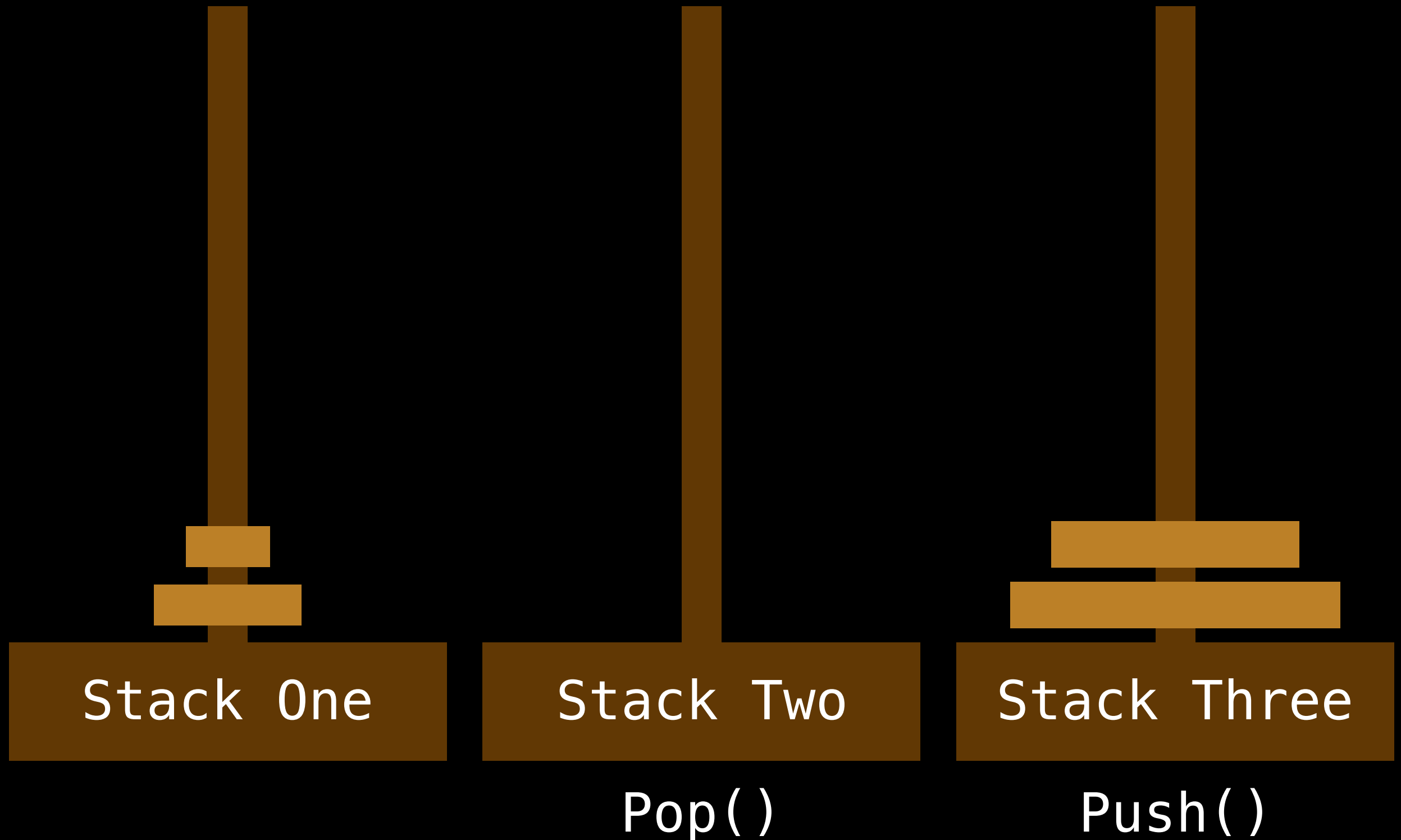
# Tower of Hanoi



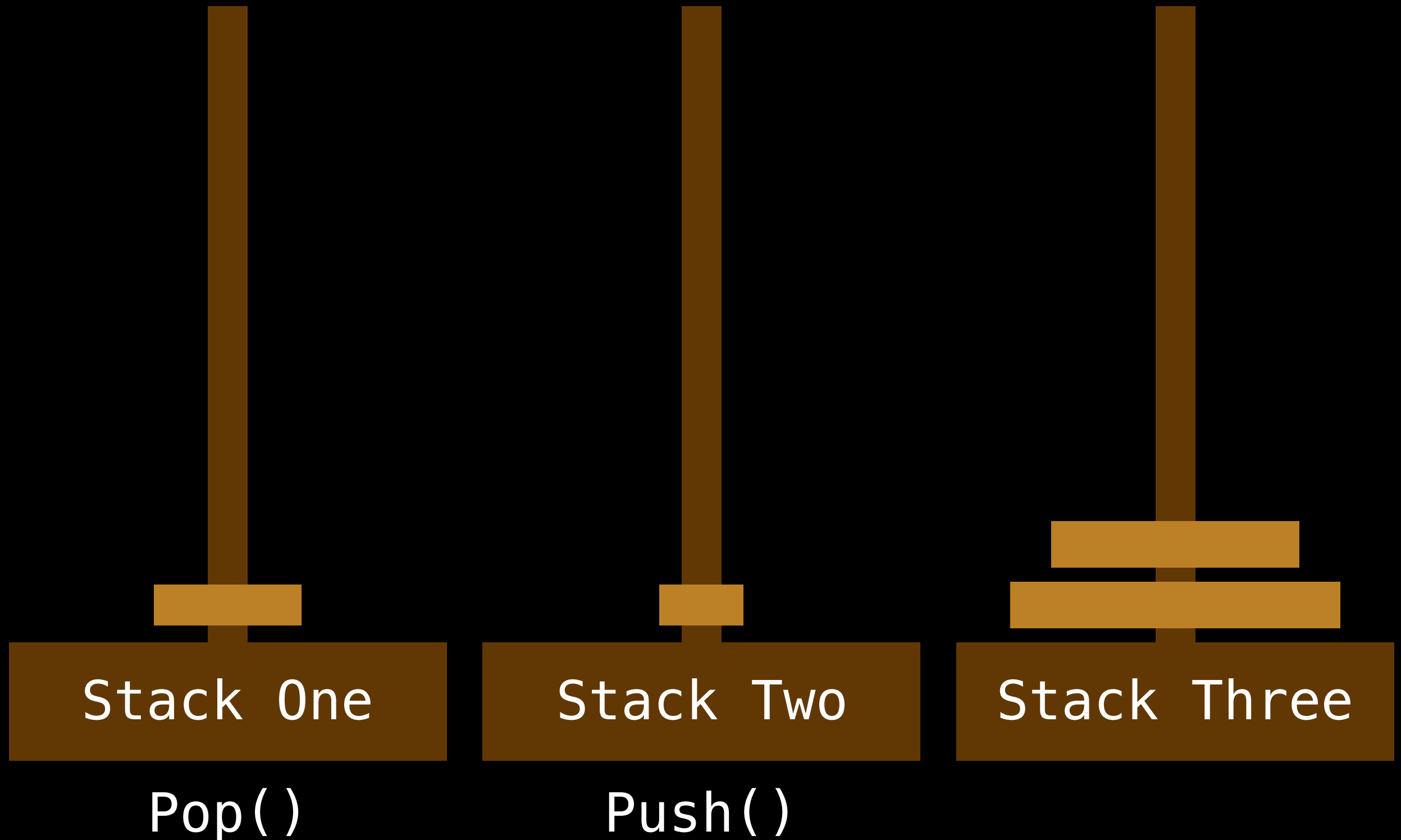
# Tower of Hanoi



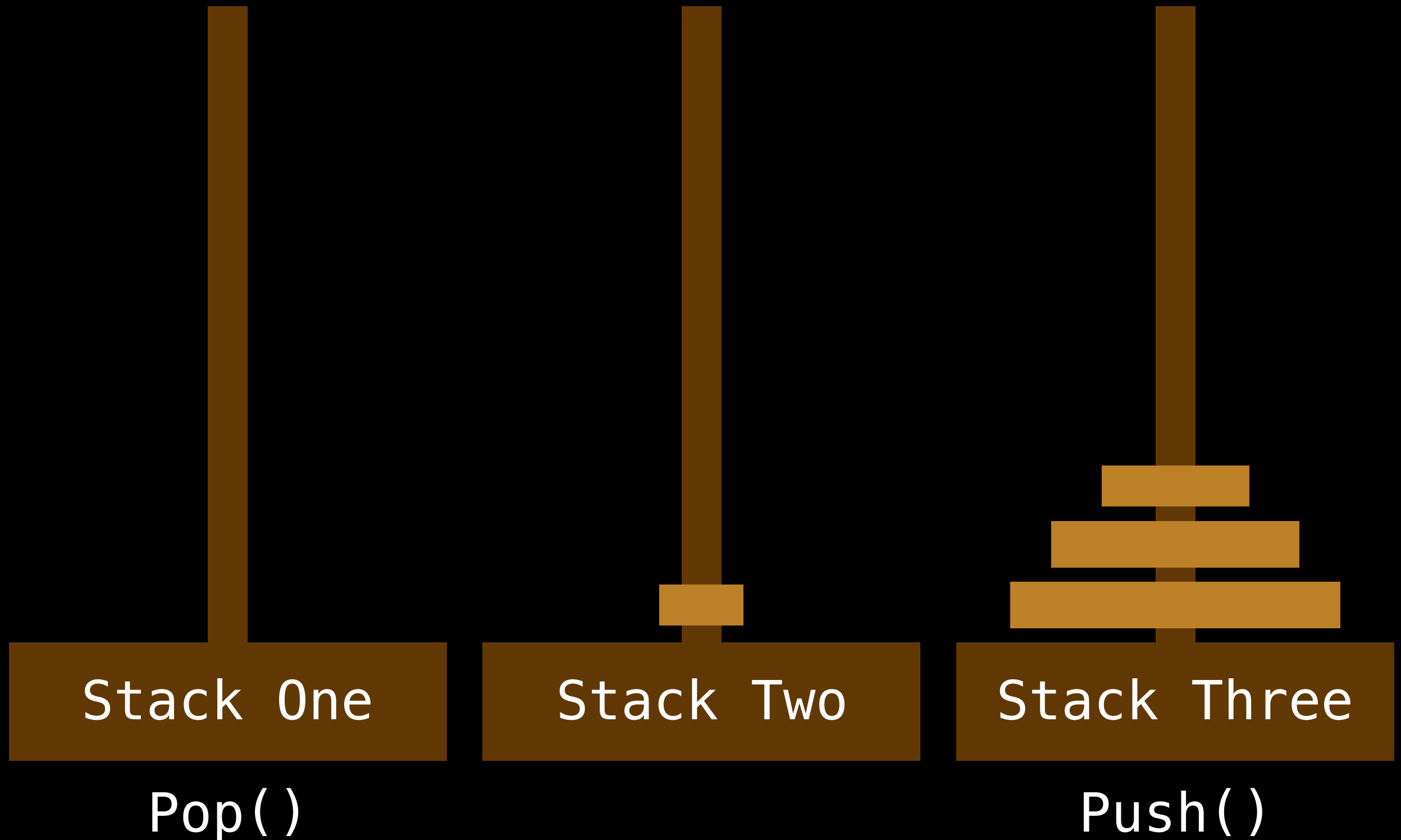
# Tower of Hanoi



# Tower of Hanoi

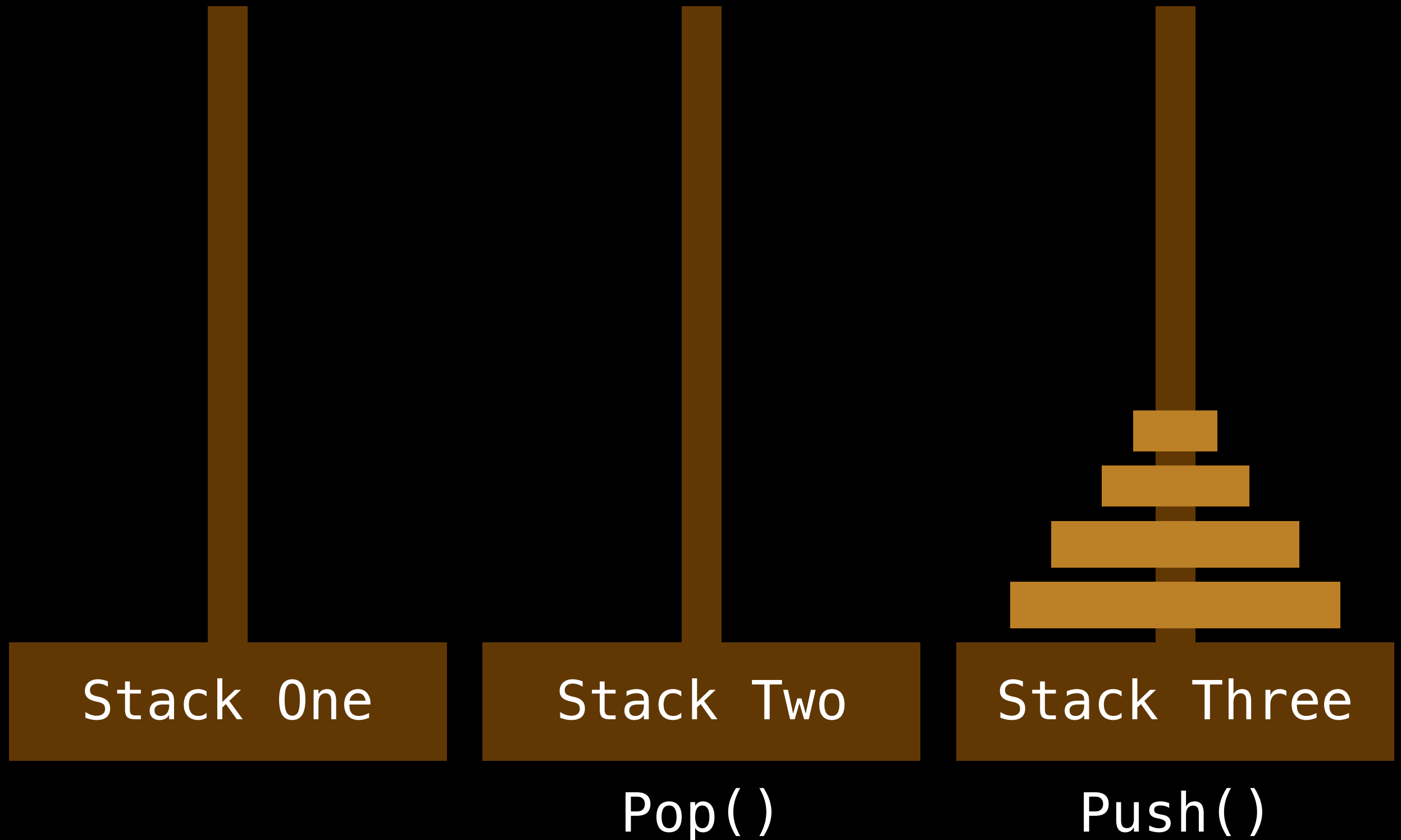


# Tower of Hanoi

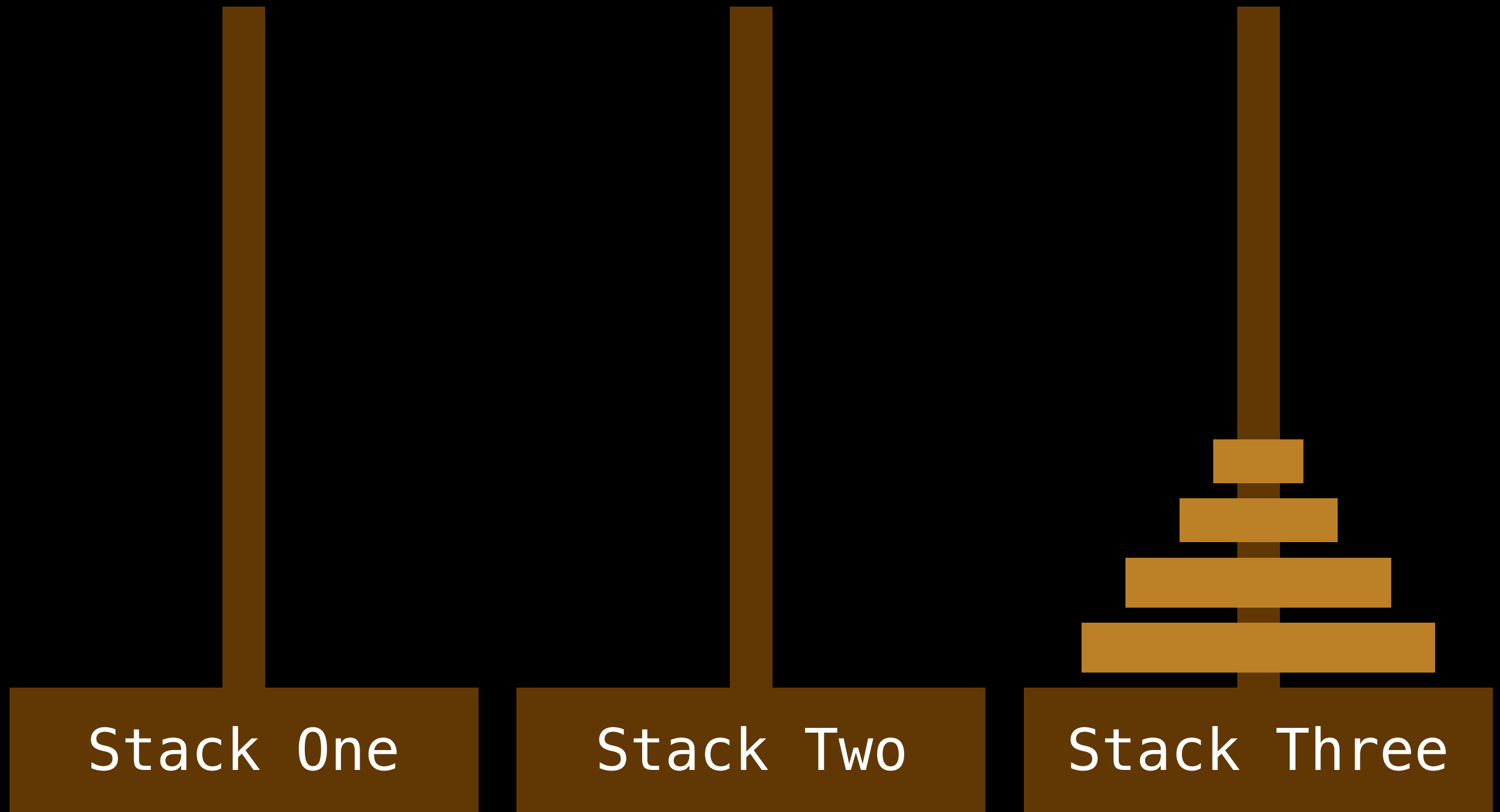




# Tower of Hanoi



# Tower of Hanoi



# Stack Operations

William Fiset

# Pushing

## Instructions

Push(4)

Push(2)

Push(5)

Push(13)

# Pushing

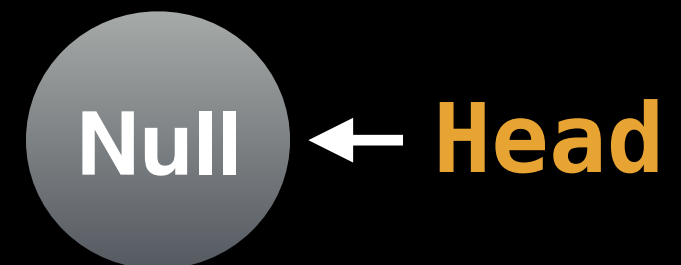
## Instructions

Push(4)

Push(2)

Push(5)

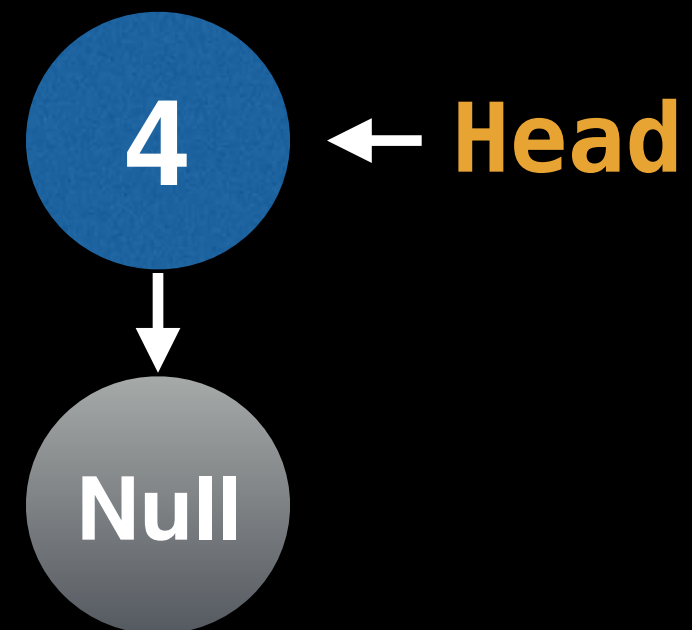
Push(13)



# Pushing

## Instructions

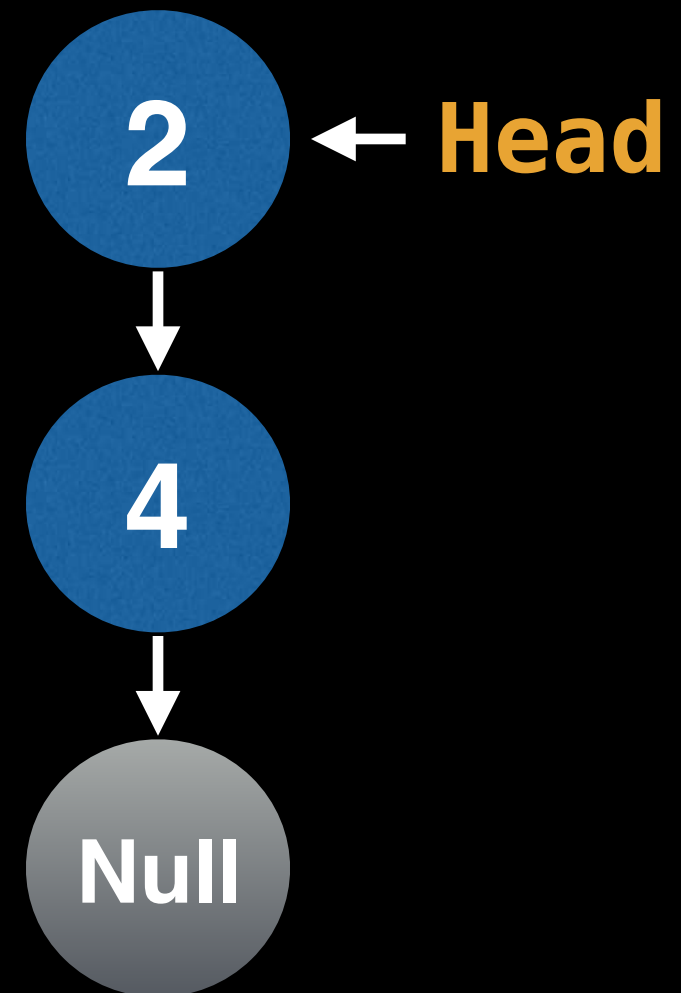
→ Push(4)  
Push(2)  
Push(5)  
Push(13)



# Pushing

## Instructions

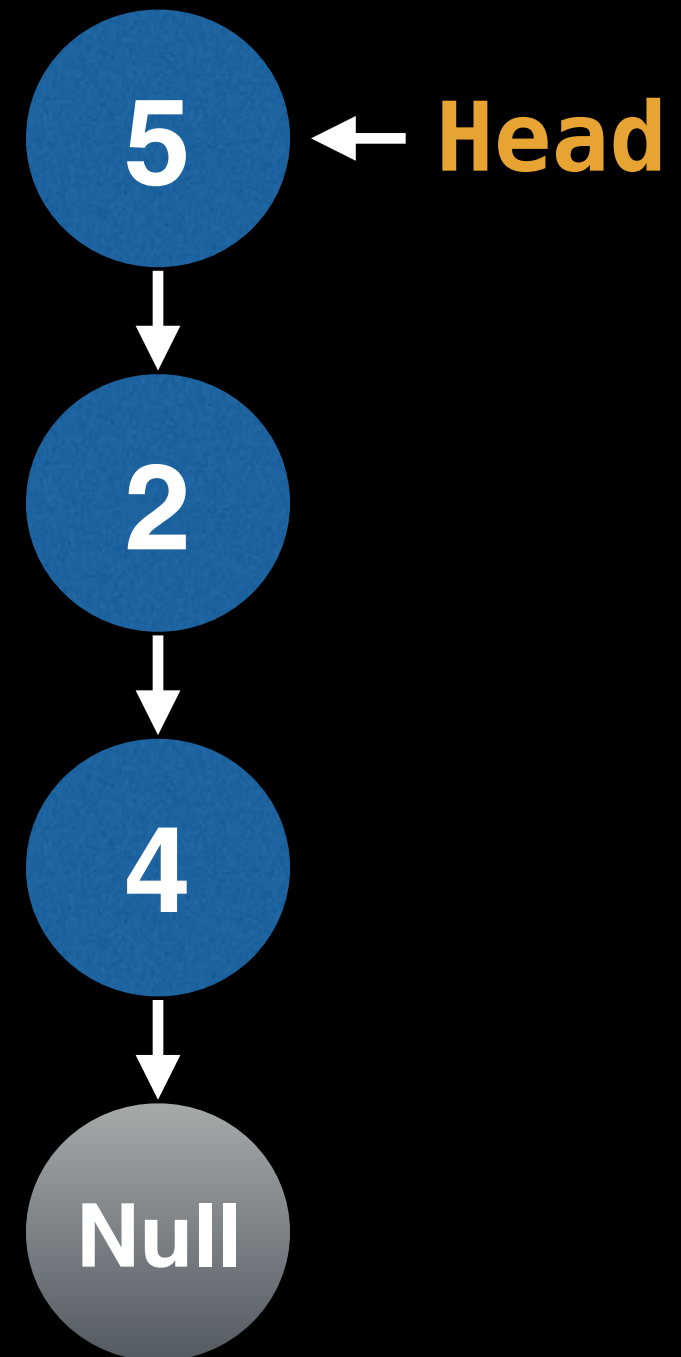
→ Push(4)  
Push(2)  
Push(5)  
Push(13)



# Pushing

## Instructions

→ Push(4)  
Push(2)  
Push(5)  
Push(13)





# Pushing

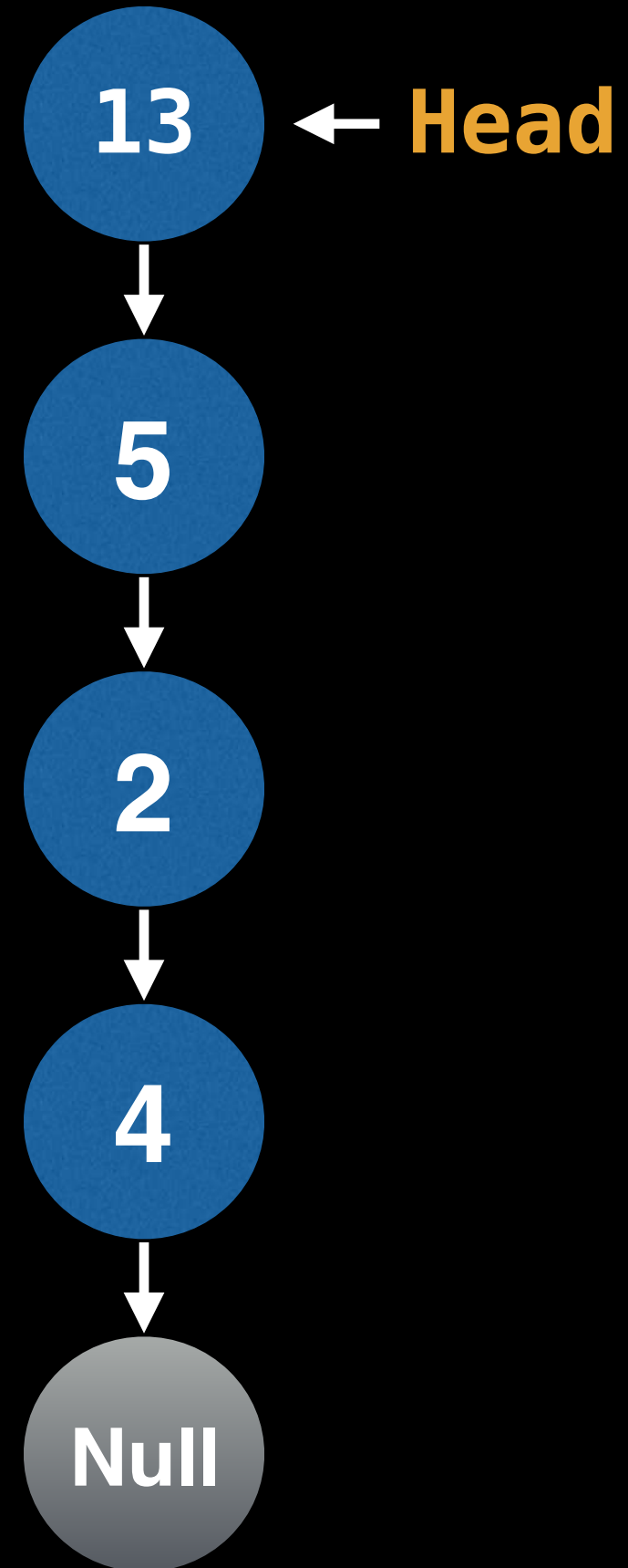
## Instructions

Push(4)

Push(2)

Push(5)

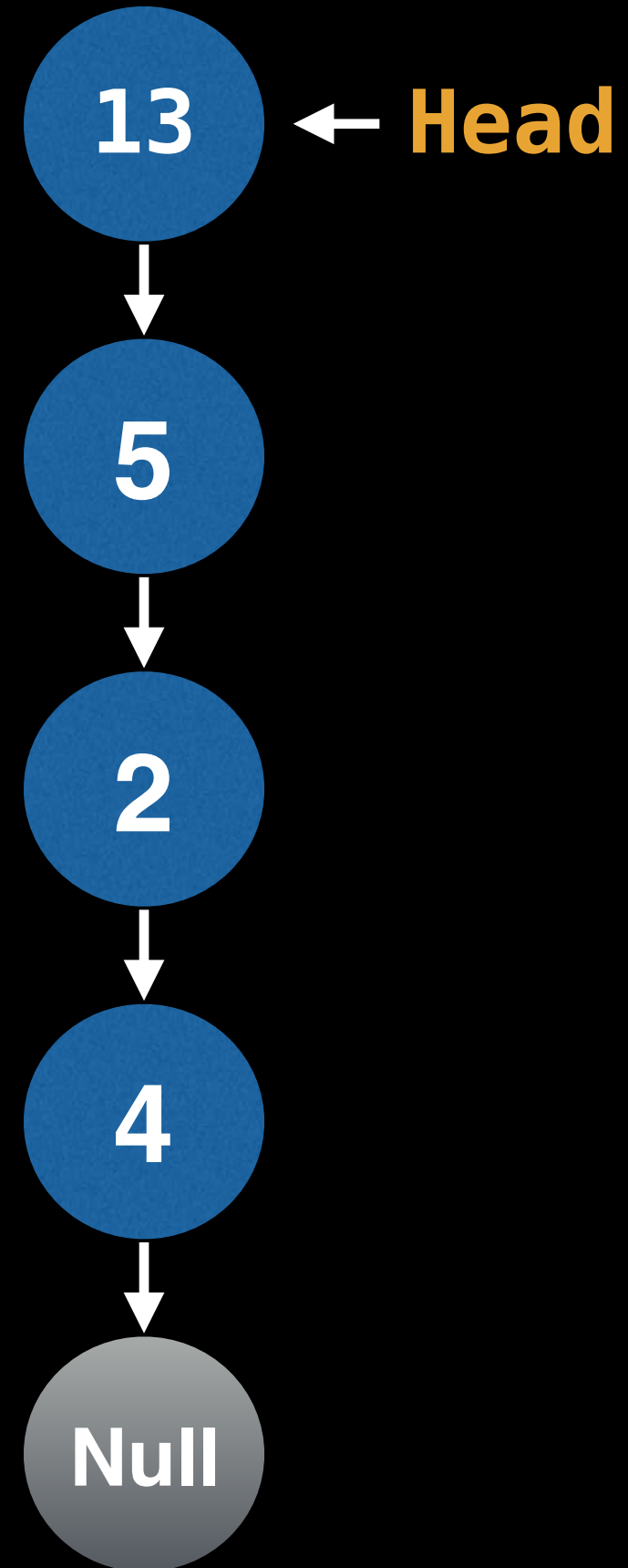
→ Push(13)



# Popping

## Instructions

Pop()  
Pop()  
Pop()  
Pop()



# Popping

## Instructions

→ Pop()  
Pop()  
Pop()  
Pop()

Null

5

← Head

2

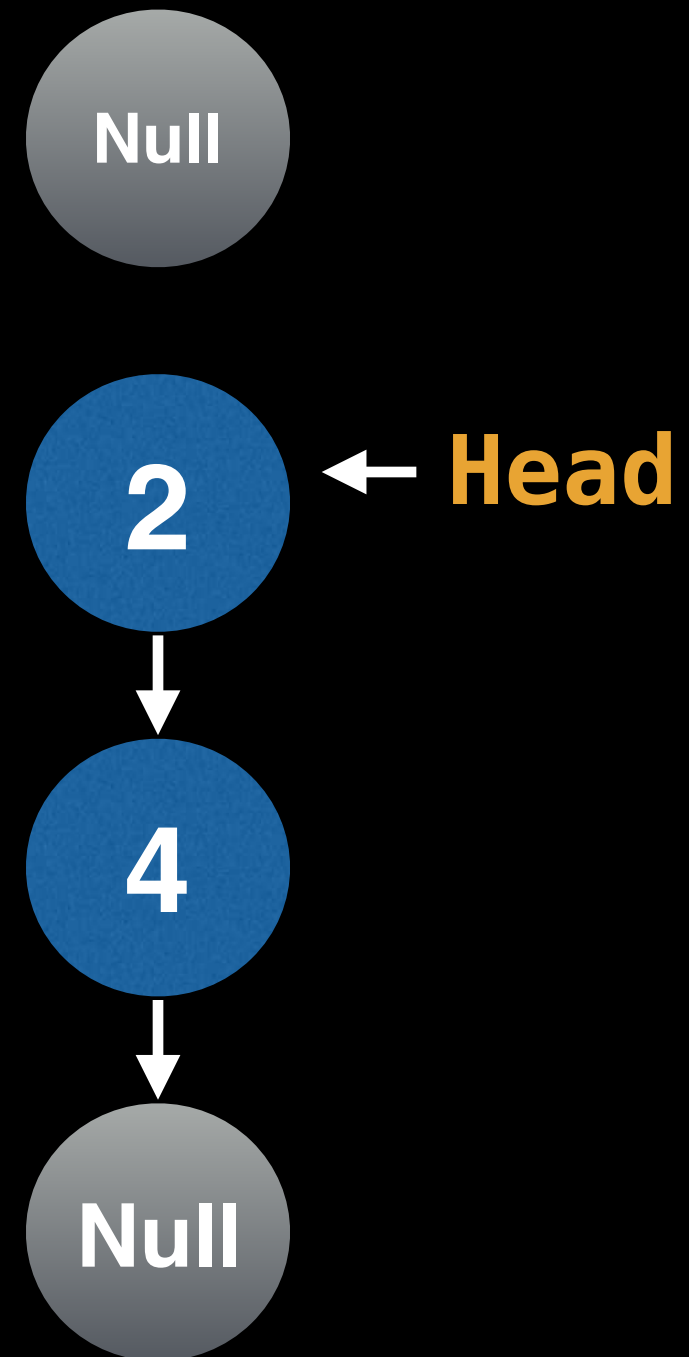
4

Null

# Popping

## Instructions

→ Pop()  
Pop()  
Pop()  
Pop()



# Popping

## Instructions

Pop()  
Pop()  
→ Pop()  
Pop()

Null

4

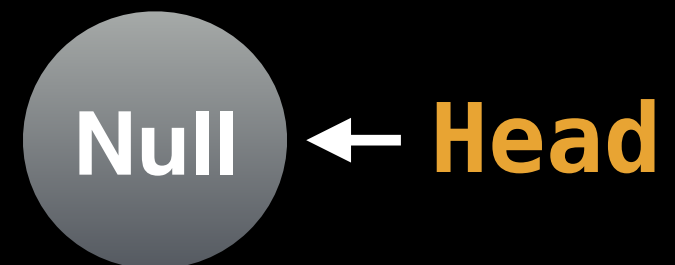
← Head

Null

# Popping

## Instructions

Pop()  
Pop()  
Pop()  
→ Pop()



# Popping

## Instructions

Pop()

Pop()

Pop()

Pop()

