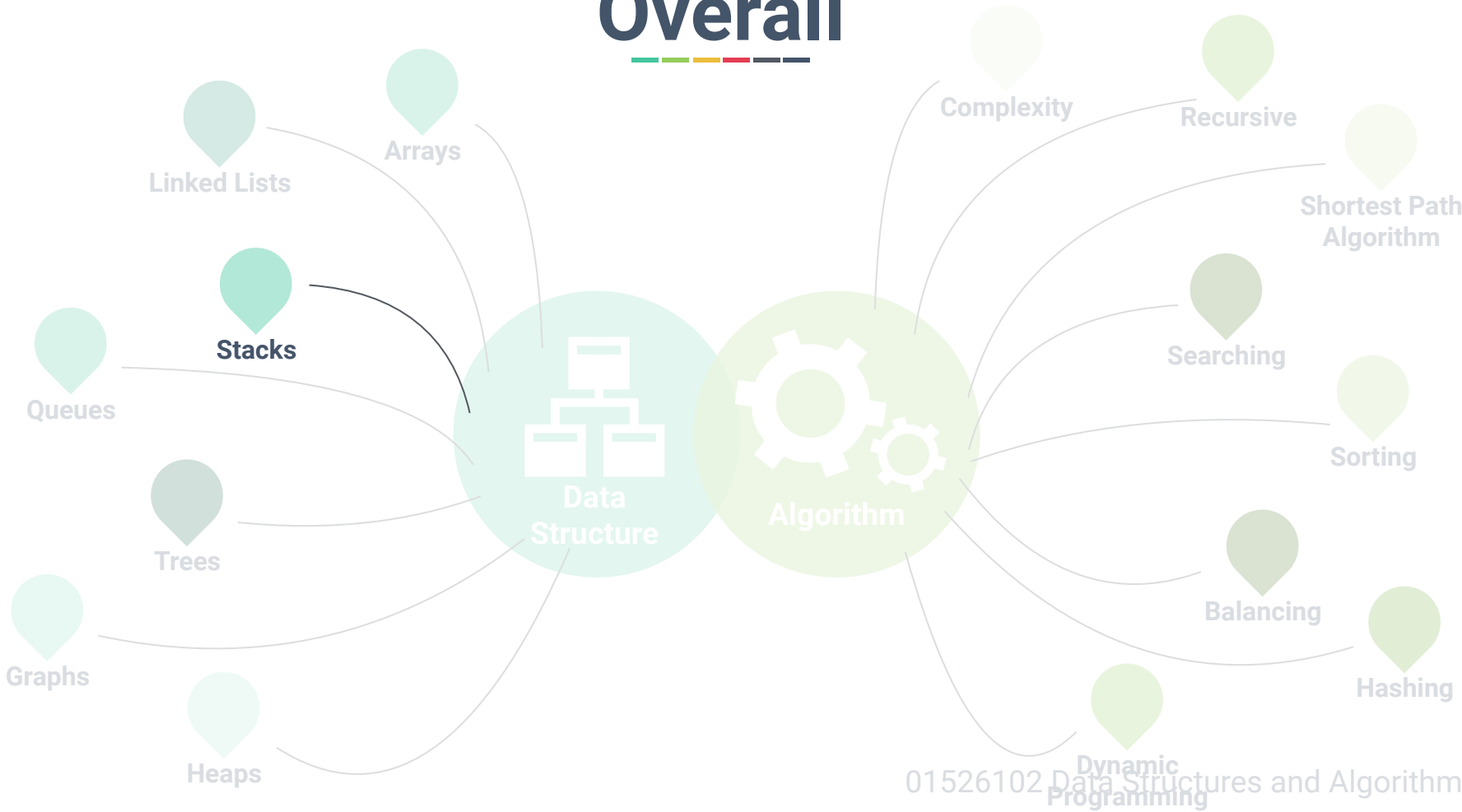
Chapter 3: Stacks

Dr. Sirasit Lochanachit

Overall


Linked Lists
Arrays
Stacks
Queues
Trees
Graphs
Heaps
Data Structure
Algorithm
Complexity
Recursive
Shortest Path Algorithm
Searching
Sorting
Balancing
Hashing
Dynamic Programming

Today's Outline

1. What is a Stack?
2. Stack Abstract Data Type
3. Simple Array-Based Stack Implementation
4. Stack Applications

Linear Data Structures

- Items are ordered depending on how they are added or removed.
  - Arrays
  - Stacks
  - Queues

What is a Stack?

A stack is a collection of objects that are inserted and removed according to the ***last-in first-out*** **(*LIFO*)** principle [1].

The name “stack” is derived from the metaphor of a stack of plates in a plate dispenser.

[1] Michael T. Goodrich et al., Data Structures and Algorithms in Python, 2013

Stack of Books

Top of stack (Accessible)

Bottom of stack (Inaccessible)

Stack Abstract Data Type

To create an instance of a stack, use Stack().

Formally, there are 2 main operations of stacks:

1) push(item)

2) pop()

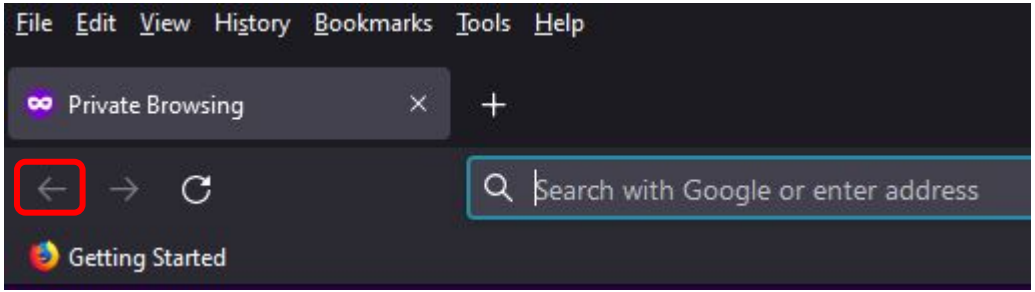
[1] Michael T. Goodrich et al., Data Structures and Algorithms in Python, 2013

Stack of Books

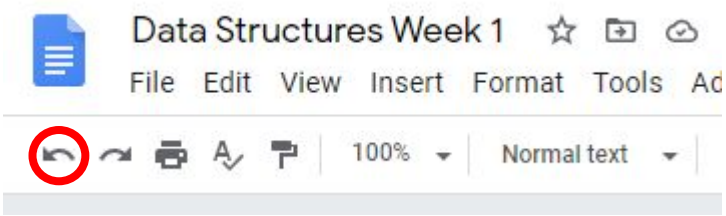
Push
a new book on the top

Pop
a book from the top

Stack Applications Example



Web Browser's history of recently visited sites.



The undo button.

Stack's Operations

Additional operations of stacks:

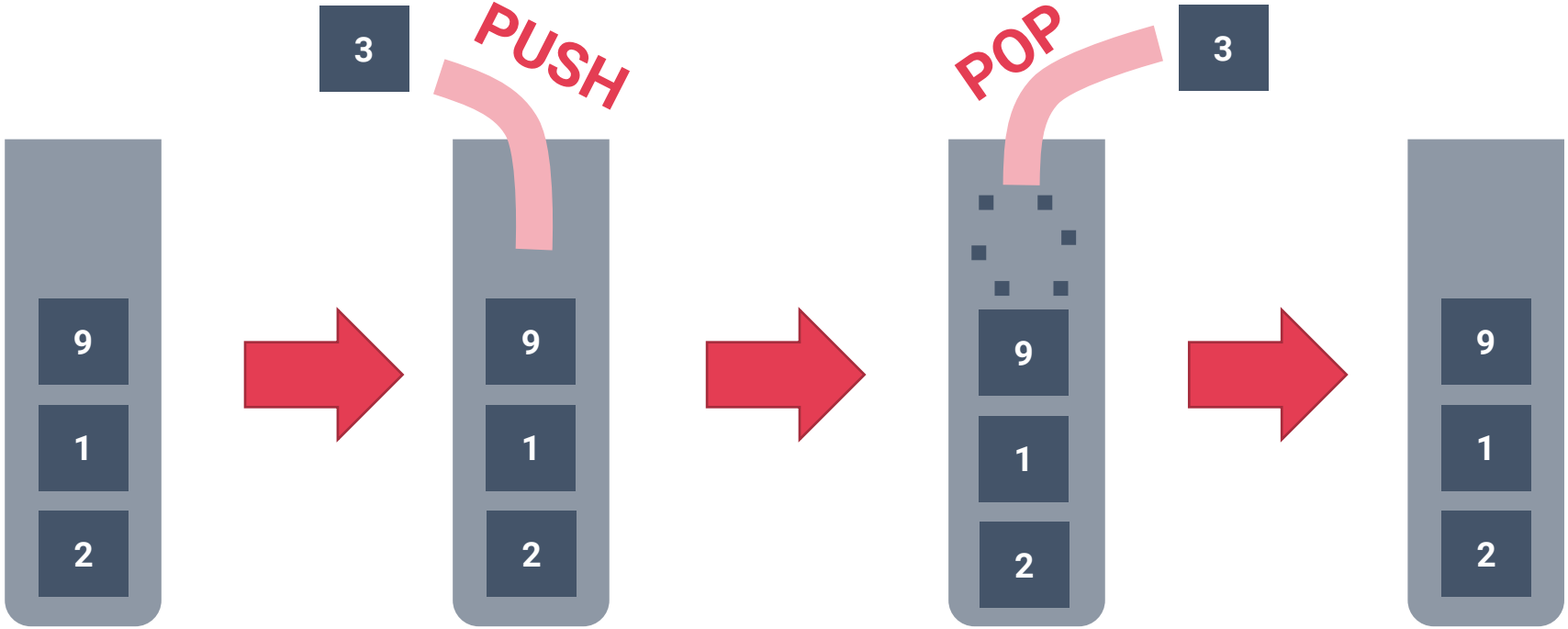
1) top()
2) is_empty()
3) len(Stack)

[1] Michael T. Goodrich et al., Data Structures and Algorithms in Python, 2013

Stack Implementation

How to Implement a Stack?

Stack Implementation



Stack's Operation Example

Operation	Return Value
S.push(9)	
S.push(5)	
S.top()	
S.push(2)	
S.pop()	
S.is_empty()	
len(S)	

Stack

Asymptotic Performance

Operation	Running Time
S.push(element)	$O(1)$
S.pop()	$O(1)$
S.top()	$O(1)$
S.is_empty()	$O(1)$
len(S)	$O(1)$

Stack Implementation

How to Implement a Stack using a Python?

Stack Implementation

Create ArrayStack class and its methods

[2]
```
class ArrayStack:
  def __init__(self):
    """ Create an empty stack. """
    self._data = []   # Initiate a nonpublic list instance

  def __len__(self):
    """ Return the number of elements in the stack. """
    return len(self._data)

  def is_empty(self):
    """ Return True if the stack is empty. """
    return len(self._data) == 0

  def push(self, element):
    """ Add an element to the top of the stack. """
    self._data.append(element) # new item stored at end of list

```

Stack Implementation

```
18  def top(self):
19    """ Return (but do not remove) the element at the top of the stack.
20    Raise an exception if the stack is empty. """
21    if self.is_empty():
22      print('Stack is empty')
23      raise Empty('Stack is empty') # Calling subclass Empty
24    return self._data[-1] # the last item in the list
25
26  def pop(self):
27    """ Remove and return the element from the top of the stack.
28    Raise an exception if the stack is empty. """
29    if self.is_empty():
30      print('Stack is empty')
31      raise Exception('Stack is empty') # Alternate way to call subclass Empty
32    return self._data.pop() # remove last item from the list
33
```

```
[3]  1 class Empty(Exception):
     2   """ Error attempting to access an element from an empty container. """
     3   pass
```

Stack Applications

1) Simple Balanced Parentheses

Each opening symbol must match its corresponding closing symbol.

Parentheses:

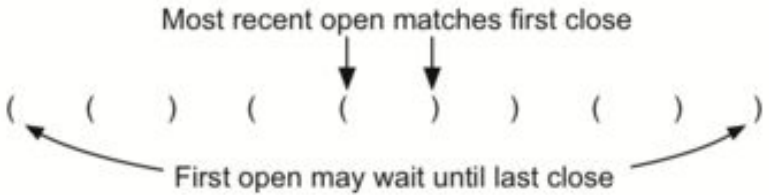
Examples

Balanced:

Unbalanced:

Simple Balanced Parentheses

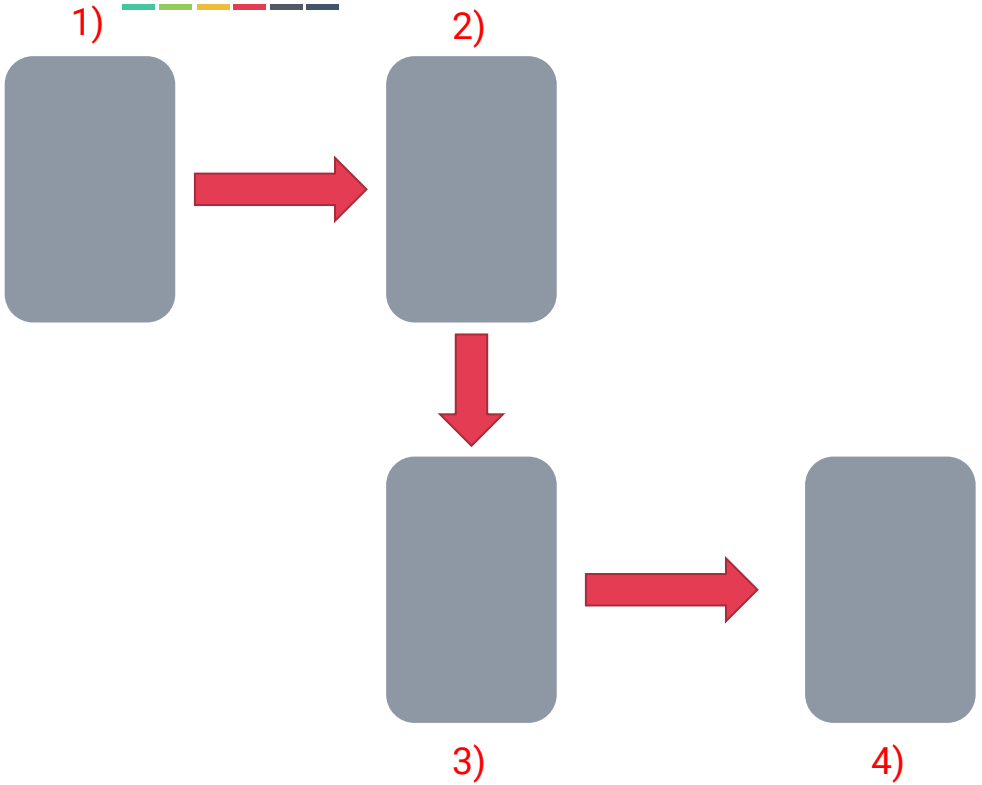
How to check balanced parentheses?



Ref: https://runestone.academy/runestone/books/published/pythonds/BasicDS/SimpleBalancedParentheses.html

Simple Balanced Parentheses

Example: (1 + x - (y + z))



Balanced Parentheses Algorithm

IsBalanced(str):

Stack Applications

2) Balanced Grouping Symbols

Each opening symbol must match its corresponding closing symbol.

Parentheses:
Brackets:
Braces:

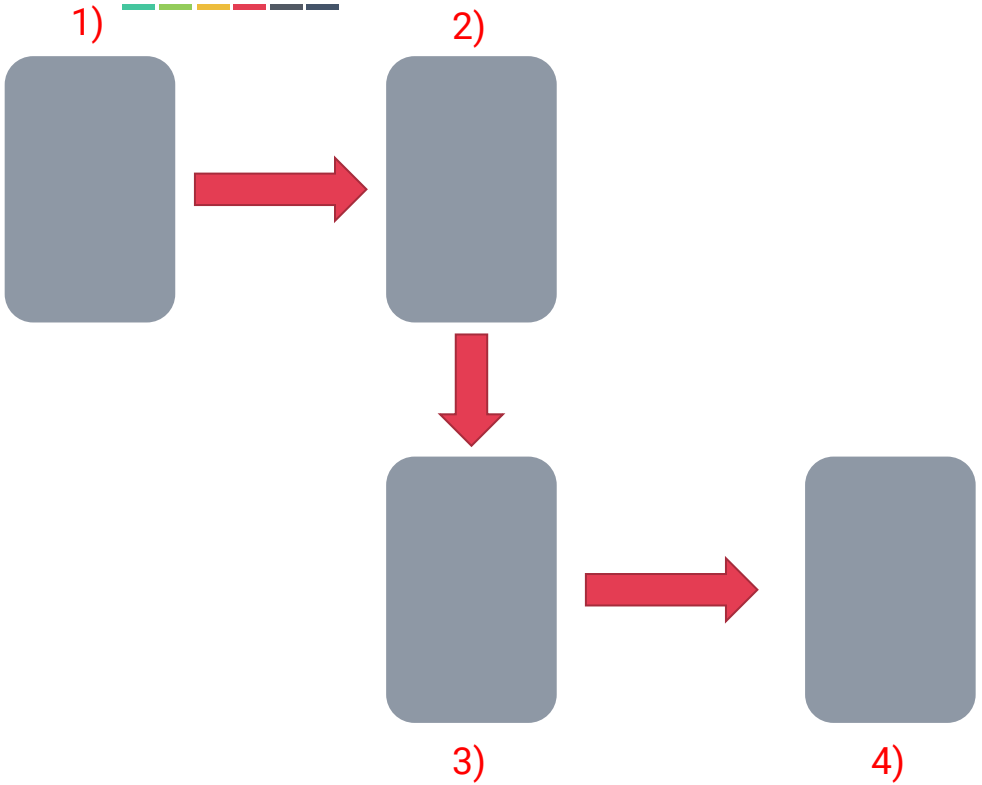
Examples

Balanced:

Unbalanced:

Balanced Grouping Symbols

Example: (1 + x - [y + z])



Balanced Symbols Algorithm

IsBalanced(str):

```
s = ArrayStack()
    ...
    ...
    ...
    ...
    ...
return s.is_empty()
```