Stacks and Abstract Data Types

CSC148, Introduction to Computer Science
Fall 2016
Diane Horton
What do these have in common?

Problem 1: Given the duration of flights between cities around the world, find the route from city A to city B with the least overall flight time.

Problem 2: Given the cost of flights between cities around the world, find the route from city A to city B with the least overall cost.
Problem 3: Given the air miles earned for flights between cities around the world, find the route from city A to city B with the most overall air miles.

Problem 4: Given the A&S calendar and timetable, plan when to take your courses in order to take a certain course as soon as possible.

Problem 5: Given the rules of a game, find the fastest way to win.
Abstract Data Type

• An **Abstract Data Type** (or ADT)
  – Defines data and operations on it.
  – Does not say anything about implementation.

• Is like the interface for a class, but so general that it transcends a programming language.

• Example ADTs:
  – Dictionary (but not specifically the Python version)
  – Priority Queue
  – Stack

• If a language doesn’t support an ADT, you can build it yourself.
The Stack ADT

• Data: 0 or more items
• Operations:
  – push
  – pop
  – check if empty
• You *don’t* need to know the implementation in order to use a Stack.
Uses for Stacks

• Function or method calls!
  
  ```python
  def fun1(x):
    return fun2(x + 5)
  
  def fun2(y):
    return 2 * fun3(y)
  
  def fun3(z):
    return z % 3
  ```

  Our frames are on a stack.

• Balancing parentheses.
a
b
b
Exercise

def is_balanced(line):
    """Return True iff <line> contains balanced brackets. Accept round, square, and curly brackets."

    @type line: str
    @rtype: bool

    >>> is_balanced('[a * (3 + b)]')
    True
    >>> is_balanced('[a * (3 + b])']
    False
    >>> is_balanced('3 - [x']
    False
    """
def size(s):
    """Return the number of items in s. Do not mutate s.
    """
    @type s: Stack
    @rtype: int

    >>> s = Stack()
    >>> size(s)
    0
    >>> s.push('hi')
    >>> s.push('more')
    >>> s.push('stuff')
    >>> size(s)
    3
    """