Ch21-Stacks

August 7, 2020

1 Stacks

http://openbookproject.net/thinkcs/python/english3e/stacks.html

- container adapters or abstract data type (ADT) that may use list or linked-list as containers to hold data
- specifically designed to operate as a LIFO (last-in-first-out) or FILO (first-in-last-out) data structure
  - last item added is the first to be removed
- built-in alternative: deque - https://docs.python.org/3/library/collections.html#collections.deque

1.1 The Stack ADT

- an ADT is defined by the operations that can be performed on it, called an interface.
- interface of stack consists of the following basic operations:
  - ___init___ - initialize a new empty stack
  - ___len___ - returns length/size of the stack
  - push - add a new item to the stack
  - pop - remove and return last item that was added to the stack
  - is_empty - check if the stack is empty

1.2 Implementing stack with Python list

- Python list provides a several methods that we can take advantage of to emulate Stack ADT

[2]:
```
class Stack:
    def __init__(self):
        self.items = []

    def __len__(self):
        return len(self.items)

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def is_empty(self):
```

```
        return len(self.items == 0)
```

1.3 Applications of stack

[3]:
```
s = Stack()
s.push(54)
s.push(45)
s.push('+')
```

1.4 visualize stack with pythontutor

https://goo.gl/Q4wZaL

[5]:
```
from IPython.display import IFrame
src = """
http://pythontutor.com/iframe-embed.
 ↪html#code=class%20Stack%3A%0A%20%20%20%20def%20__init__%28self%29%3A%0A%20%20%20%20%20%20%2
 ↪items%20%3D%20%5B%5D%0A%20%20%20%20%0A%20%20%20%20def%20__len__%28self%29%3A%0A%20%20%20%20
 ↪items%29%0A%20%20%20%20%0A%20%20%20%20def%20push%28self,%20item%29%3A%0A%20%20%20%20%20%20%
 ↪items.
 ↪append%28item%29%0A%20%20%20%20%20%20%20%20%0A%20%20%20%20def%20pop%28self%29%3A%0A%20%20%2
 ↪items.
 ↪pop%28%29%0A%20%20%20%20%0A%20%20%20%20def%20is_empty%28self%29%3A%0A%20%20%20%20%20%20%20%
 ↪items%20%3D%3D%200%29%0A%20%20%20%20%20%20%20%20%0As%20%3D%20Stack%28%29%0As.
 ↪push%2854%29%0As.push%2845%29%0As.
 ↪push%28'%2B'%29&codeDivHeight=400&codeDivWidth=350&cumulative=false&curInstr=0&heapPrimitiv
 ↪js&py=3&rawInputLstJSON=%5B%5D&textReferences=false
"""
IFrame(src, width=900, height=500)
```

[5]: <IPython.lib.display.IFrame at 0x1049d97f0>

1.5 using a stack to evaluate postfix notation

- infix notation: 1 + 2
- prefix notation: + 1 2
- postfix notation: 1 2 +
  - operator follows the operands
  - no need to use parenthesis to control oder of operations
- algorithm steps:
  1. starting at the beginning of the expression, get one term/token (operator or operand) at a time
     a. if the term is an operand, push it on the stack
     b. if the term is an operator, pop two operands off the stack, perform the operation on them, and push the result back on the stack
  2. When you get to the end of the expression, there should be exactly one operand left on the stack, the result.

[8]:
```
# given a postfix notation such as: 56 47 + 2 *, the following function␣
↪evaluates the result using Stack ADT
def eval_postfix(expr):
    tokens = expr.split()
    stack = Stack()
    for token in tokens:
        token = token.strip()
        if not token:
            continue
        if token == '+':
            s = stack.pop() + stack.pop()
            stack.push(s)
        elif token == '*':
            prod = stack.pop() * stack.pop()
            stack.push(prod)
        # /, and - are left as exercise
        else:
            stack.push(int(token))

    return stack.pop()
```

[9]:
```
print(eval_postfix('56 47 + 2 *'))
```

```
206
```

[11]:
```
# which is same as: (56 + 47) * 2 in infix notation
eval('(56 + 47) * 2')
```

[11]: 206

1.6 exercises

1. Write a function that converts infix notation to postfix notation, e.g., given infix notation $4 - 2 + 3$ the program should output corresponding postfix notation $4\ 2 - 3 +$

1.7 Following Kattis problems can be solved using Stack

1. Backspace problem: https://open.kattis.com/problems/backspace

- Game of throwns: https://open.kattis.com/problems/throwns
- Even Up Solitaire: https://open.kattis.com/problems/evenup
- Working at the Restaurant: https://open.kattis.com/problems/restaurant
- Pairing Socks: https://open.kattis.com/problems/pairingsocks
- Find stack-based problems in Kattis: https://cpbook.net/methodstosolve search for stack

[ ]: