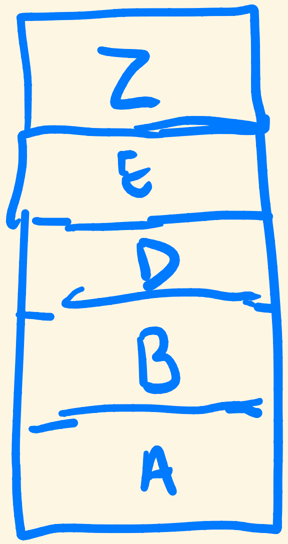
STACKS

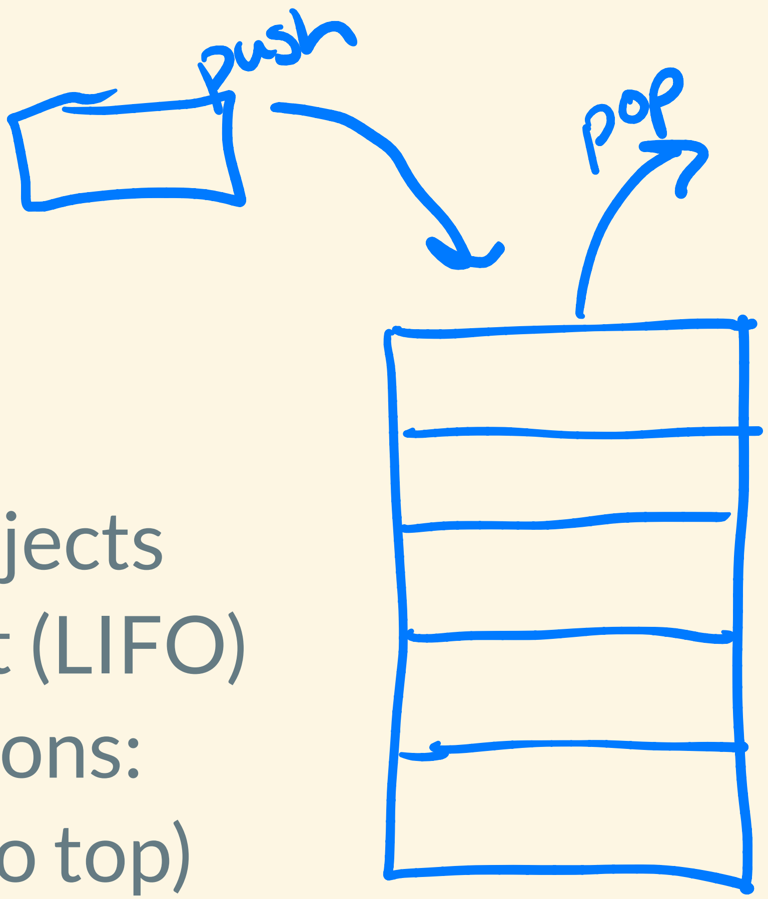
ABSTRACT DATA TYPE (ADT)

- Model of a data structure
    - NOT the actual implementation
- Describes:
    - set of data values
    - the operations that can be performed
    - what the operations do (not how they do them...)
- Language independent
    - Helpful for approach/algorithm

STACK

- Collection of objects
- Last in - first out (LIFO)
- Primary operations:
    - push (add to top)
    - pop (remove from top)





REAL LIFE EXAMPLES:

- Piles of items at grocery stores
- Shopping cart corral
- Stack of plates
- Pez dispenser

WHEN TO USE?

- Depends on the problem
- Not useful for all problems
- But -- Really useful for some problems
- Something to consider before you start coding
    - algorithm stage -- which ADT makes sense to use

$4+5 \rightarrow 9$

APPLICATION: POST-FIX NOTATION

- Infix:
  - what you're used to
  - <operand> <operator> <operand>
  - relies on order of operations and parentheses
  - Ex: 3 + 2 * 4 $3+(2*4)=3+8=11$
- Postfix:
  - <operand> <operand> <operator>
  - Ex: 3 2 4 * +   4 5 +

    3 8 +

    11

APPLICATION: POST-FIX NOTATION

- Computer has to parse math expressions
- Postfix is easier
- How could we write parser to turn expression into code?

3 2 4 * +

operators : + - * /

$2 * 4 = 8$

$3 + 8 = 11$

parse func

```
for c in input:
    if c not operator:
        push(c)
    else:
        a = pop( )
        b = pop()
        res = operate(b, a, c)
        push (res)
return (pop())
```

$\boxed{11}$

10 2 8 * + 3 -

2 * 8 = 16

10 + 16 = 26

26 - 3 = 23

$\boxed{23}$

CODING APPLICATIONS - OTHERS

- Reversing a string
- Back button in browser
- Undo/redo
- Balanced parentheses
- Maze solving
- Function call stack