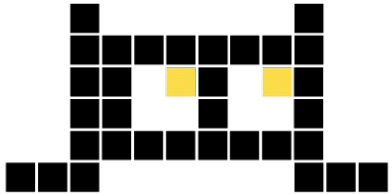
We'll Be Starting Shortly!

To help us run the workshop smoothly, kindly:

- Submit all questions using the Q&A function
- Materials at https://bit.ly/3a5t30n
- **You need a Google Account to open the .ipynb file on Google Colab. Once opened, save a copy to your Drive.**

Arrays and Stacks with Python

by



Learn to Code, Code to Learn

CODING ACADEMY

Hi!

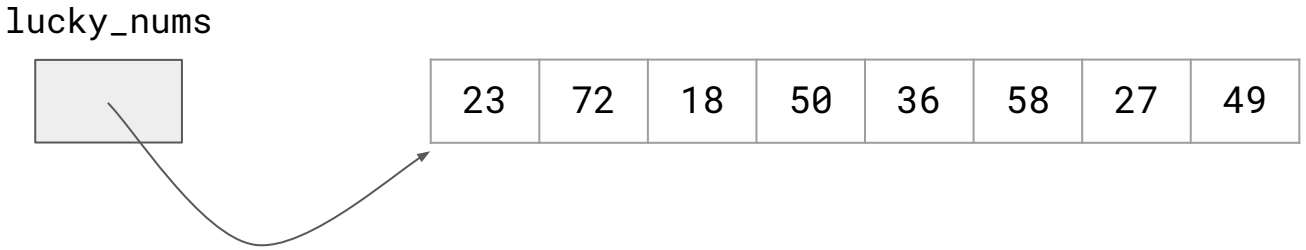
We are Melvin and Juliana
from LCCL Coding Academy in Singapore

Learn to Code, Code to Learn
CODING ACADEMY

Array

Lists in CPython are implemented with arrays, a contiguous section of memory.

```
lucky_nums = [23, 72, 18, 50, 36, 58, 27, 49]
```



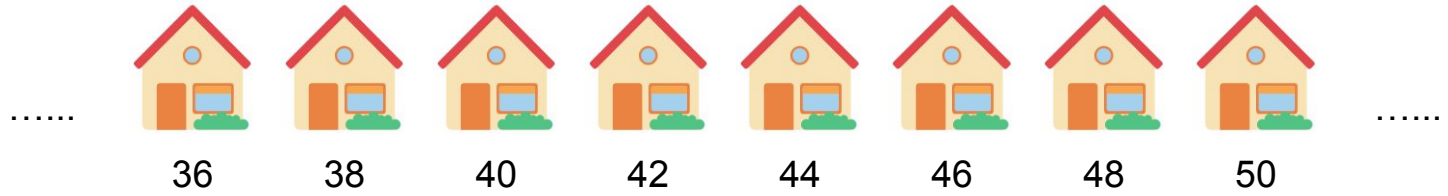
Array

Used when we need fast access to the i-th element, also supports efficient pop from the end.

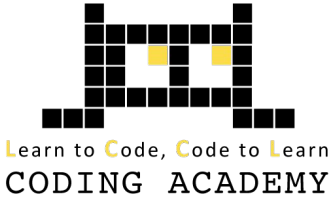
```
lucky_nums = [23, 72, 18, 50, 36, 58, 27, 49]
```

0	1	2	3	4	5	6	7
23	72	18	50	36	58	27	49

Array



Hands-On - Array Experiment

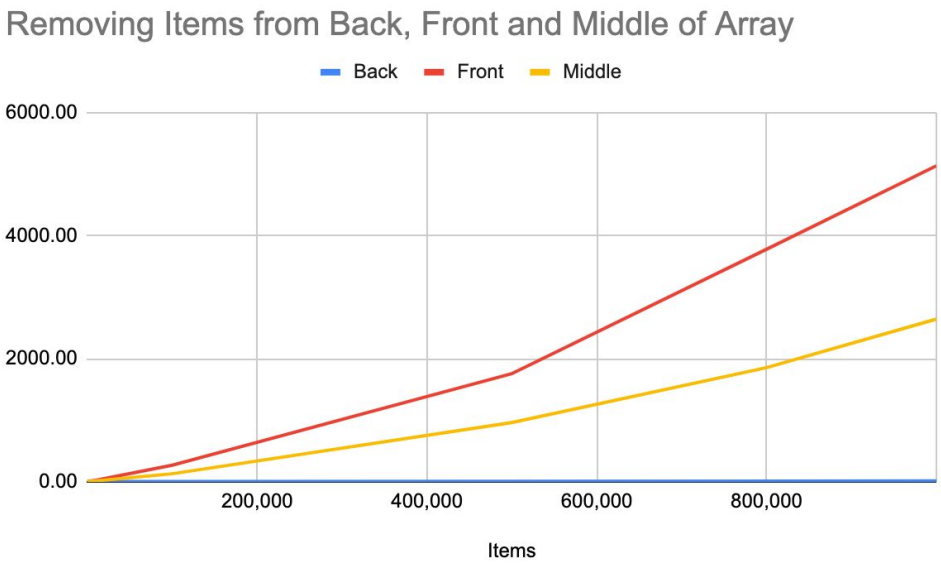


Compare the time needed to:

- remove items from the back vs
- removing items from the front vs
- removing from the middle

for lists of lengths 10, 100, 1000, 10000.

Array Experiment



Items	Back (e-7 s)	Front (e-7 s)	Middle (e-7 s)
10	4.82	6.63	6.29
100	5.70	7.59	7.00
1,000	5.09	8.21	8.64
10,000	5.81	24.69	16.96
100,000	6.13	269.74	131.30
500,000	9.42	1756.77	962.00
800,000	11.20	3777.56	1855.00
1,000,000	14.40	5134.00	2644.00

Array Experiment - What Happened

array

23	72	18	50	36	58	27	49

Removing from the back:

array

23	72	18	50	36	58	27

Array Experiment - What Happened

`array`

23	72	18	50	36	58	27	49

Removing from the front:

`array`

	72	18	50	36	58	27	49

Array Experiment - What Happened

`array`

23	72	18	50	36	58	27	49

Removing from the front:

`array`

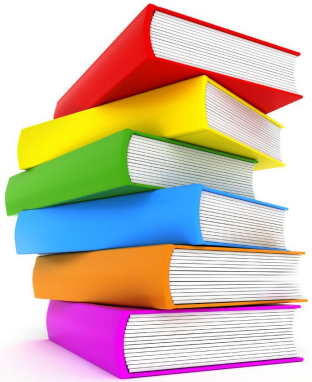
72	18	50	36	58	27	49

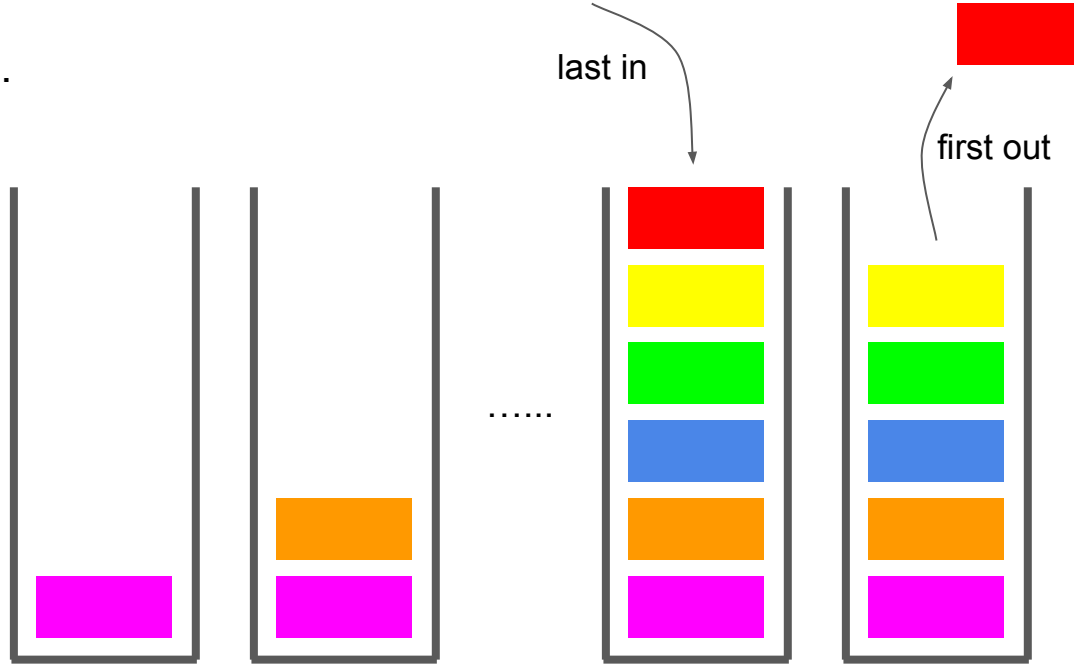
Stack

Stack is a data type which allows adding to the top (push) and removing from the top (pop). It can be implemented using list.

Stack

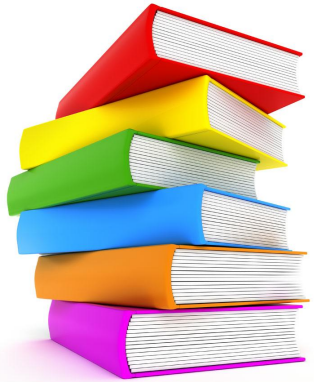
Last in, first out (LIFO).





Stack

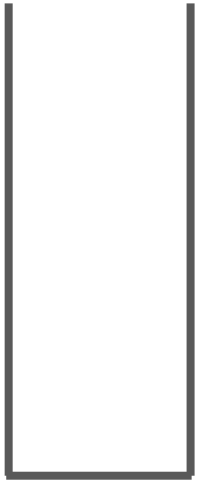
Using Python list (array) to model a stack:



```
books = []
books.append('p')    # ['p']
books.append('o')    # ['p', 'o']
books.append('b')    # ['p', 'o', 'b']
books.append('g')    # ['p', 'o', 'b', 'g']
books.append('y')    # ['p', 'o', 'b', 'g', 'y']
books.append('r')    # ['p', 'o', 'b', 'g', 'y', 'r']
books.pop()          # ['p', 'o', 'b', 'g', 'y']
books.pop()          # ['p', 'o', 'b', 'g']
books.pop()          # ['p', 'o', 'b']
books.pop()          # ['p', 'o']
books.pop()          # ['p']
books.pop()          # []
```

Stack Example 1 - Delimiter Soup

Delimiter soup: https://open.kattis.com/problems/delimitersoup



```
([] [] ]
```

Stack Example 1 - Delimiter Soup


```
stack = []
opening = {']': '[', '}': '{', ')': '('}

for i, c in enumerate(sequence):
    if c == ' ':
        pass
    elif c in opening.values(): # if open delim
        stack.append(c)         # push to stack
    # must be closing delim
    elif len(stack) == 0 or opening[c] != stack[-1]:
        # case where stack is empty or stack top does not match
        print(c, i)
        break
    else:
        # case where stack top matches
        stack.pop()
else:
    print('ok so far')
```

Stack Example 2 - Key Logger

Keystrokes: https://open.kattis.com/problems/lyklagangriti

What's your algorithm?

Stack Example 2 - Key Logger


```
left, right = [], []

for c in sequence:
    if c == 'L':
        right.append(left.pop())
    elif c == 'R':
        left.append(right.pop())
    elif c == 'B':
        left.pop()
    else:
        left.append(c)

right.reverse()
password = ''.join(left) + ''.join(right)
```

Learn to Code, Code to Learn
CODING ACADEMY

Shopee CODE LEAGUE 2021

Get Complimentary Video

LccLcoding.com/shopee

Your Feedback Matters!

bit.ly/3hmJ3Nr