

# Lists as Stacks and Queues



**SoftUni Team**  
**Technical Trainers**



**SoftUni**



**Software University**

<https://softuni.bg>

## 1. Stacks

- Creating Stacks
- Adding Elements
- Removing Elements

## 2. Queues

- Creating Queues
- Adding Elements
- Removing Elements



[sli.do](https://sli.do)

**#python-advanced**



**Stacks**

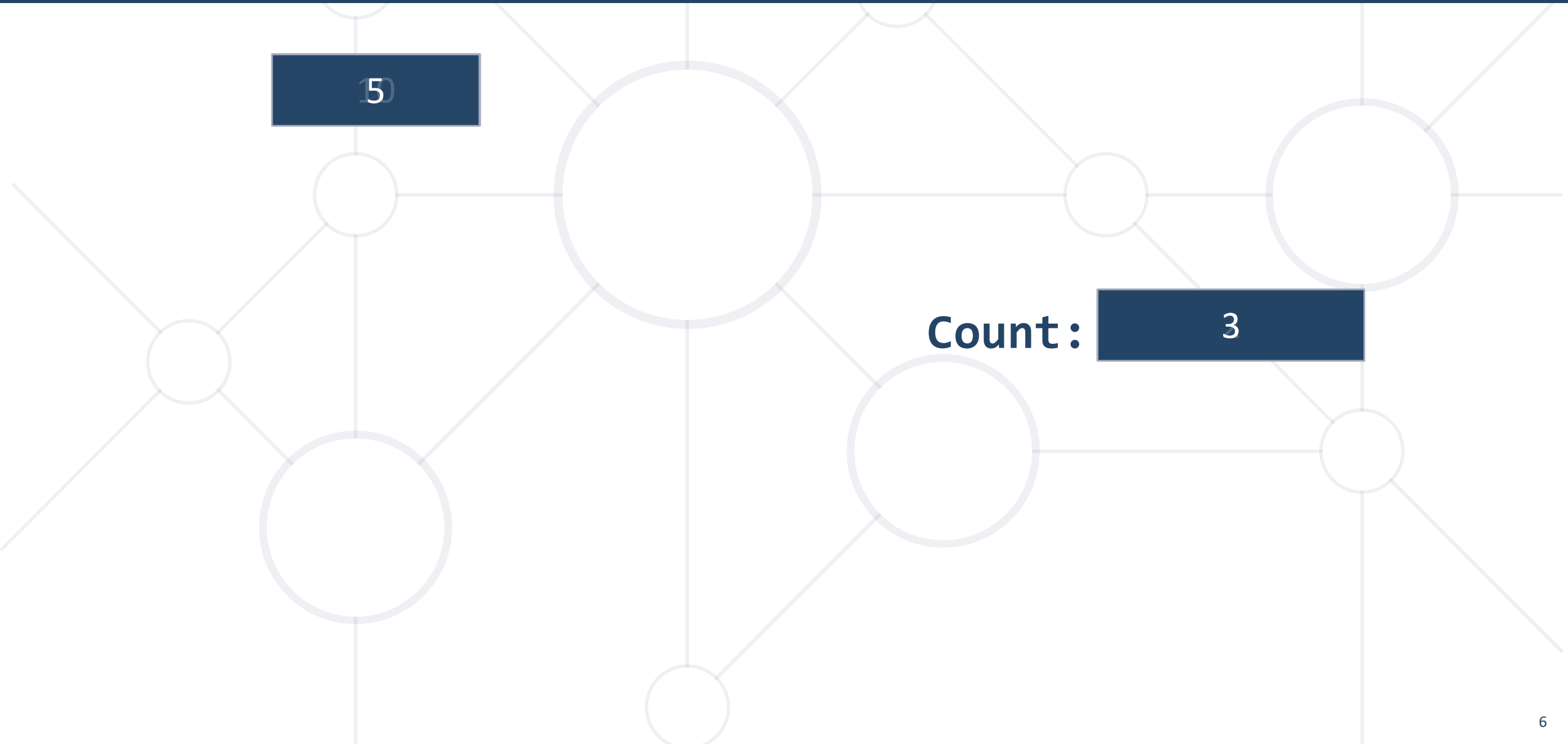
LIFO

# What is a Stack?

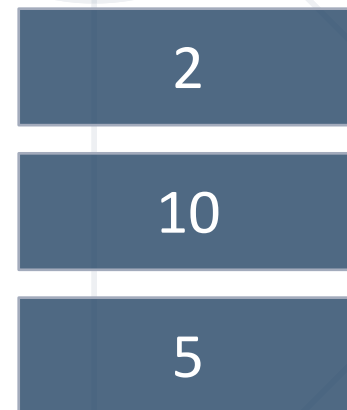
- A stack is a **linear** data structure that stores items
- The process of adding data to a stack is referred to as a "**push**"
- **Retrieving** data from a stack is called a "**pop**"
- Elements in a stack are added/removed from the top ("last in, first out") or **LIFO** order



# Push to a Stack



# Pop from a Stack



**Count:**

2

# Stacks in Python

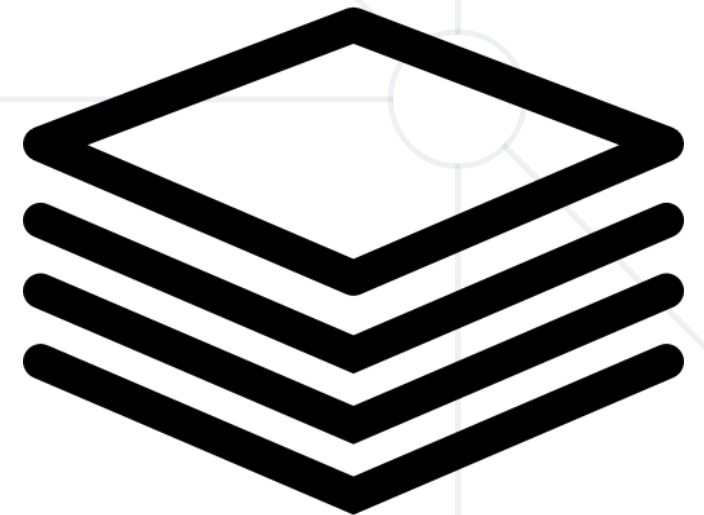
- The **list** methods make it very easy to use a list as a **stack**
- To add an item to the top of the stack, use **append()**
- To retrieve an item from the top of the stack, use **pop()**





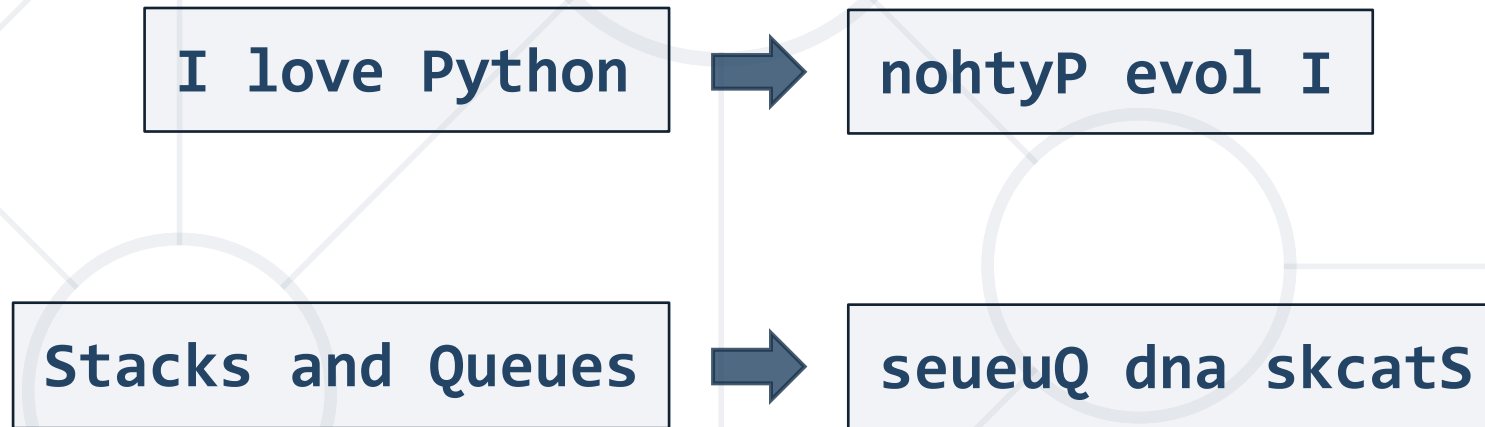
# Stacks in Python: Example

```
stack = [3, 4, 5]
stack.append(6)
stack.append(7)
print(stack) # [3, 4, 5, 6, 7]
stack.pop()  # 7
print(stack) # [3, 4, 5, 6]
stack.pop()  # 6
stack.pop()  # 5
print(stack) # [3, 4]
```



# Problem: Reverse Strings

- Write a program that reads a string, reverses it using stacks and prints the result on the console



# Solution: Reverse Strings

```
text = list(input())  
stack = []  
  
for i in range(len(text)):  
    stack.append(text.pop())  
  
print("".join(stack))
```



# Problem: Matching Parentheses

- We are given an algebraic expression with parentheses
- Scan through the string and extract each set of parentheses
- Print the result back on the console

1 + (2 - (2 + 3) \* 4 / (3 + 1)) \* 5



(2 + 3)

(3 + 1)

(2 - (2 + 3) \* 4 / (3 + 1))

# Solution: Matching Parentheses

```
text = input()
parentheses = []

for i in range(len(text)):
    if text[i] == "(":
        parentheses.append(i)
    elif text[i] == ")":
        start_index = parentheses.pop()
        print(text[start_index:i + 1])
```

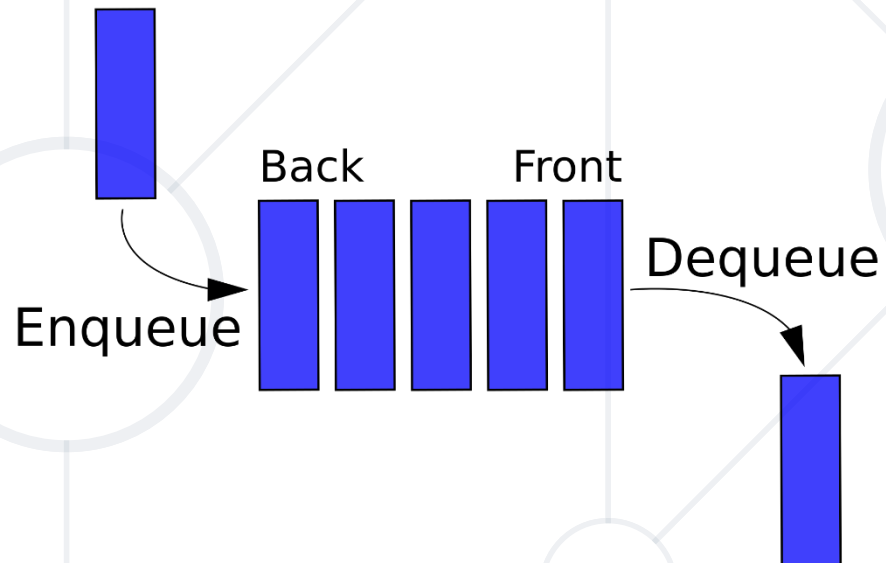


**Queues**

**FIFO**

# What is a Queue?

- A queue is a **first-in first-out** (FIFO) abstract data type
- We use them when we want things to happen in the **order** that they were **called**



# Enqueue

151

Count:

4



# Deque

**Count:** 3

121

15

-3

5

# Queues in Python

- It is possible to use a list as a queue, however they are not efficient for this purpose
- Doing inserts or pops from the beginning of a list is **slow**
- That's why we use **`collections.deque`**
- We use **`append()`** to add to the queue and **`popleft()`** to remove from the queue




# Queues in Python: Example

```
from collections import deque
queue = deque(["Eric", "John", "Michael"])
queue.append("Terry")           # Terry arrives
queue.append("Graham")         # Graham arrives
queue.popleft()                # First Leaves: 'Eric'
queue.popleft()                # Second Leaves: 'John'
print(queue)                   # ['Michael', 'Terry', 'Graham']
```

# Problem: Supermarket

- Read an input with a name and add it to a queue until **"End"**
  - If you receive **"Paid"**, print every customer and empty the queue
  - When you receive **"End"** print the remaining people in the format: **"{count} people remaining."**



Anna  
Michael  
Simone  
End

➔

3 people remaining.

# Solution: Supermarket

```
from collections import deque
queue = deque()
while True:
    command = input()
    if command == "Paid":
        while len(queue):
            print(queue.popleft())
    elif command == "End":
        print(f"{len(queue)} people remaining.")
        break
    else:
        queue.append(command)
```

# Problem: Water Dispenser

- Read the problem description from the Lab Documents and test with the provided inputs

```
2
Peter
Amy
Start
2
refill 1
1
End
```



```
Peter got water
Amy got water
0 liters left
```

# Solution: Water Dispenser

```
from collections import deque
queue = deque()
water_quantity = int(input())
name = input()
# add people to queue
command = input()
while command != "End":
    if "refill" in command:
        # increase liters
    else:
        liters = int(command)
        if liters <= water_quantity:
            water_quantity -= liters
            print(f"{queue.popleft()} got water")
        else:
            print(f'{queue.popleft()} must wait')
    command = input()
print(f"{water_quantity} liters left")
```



# Practice

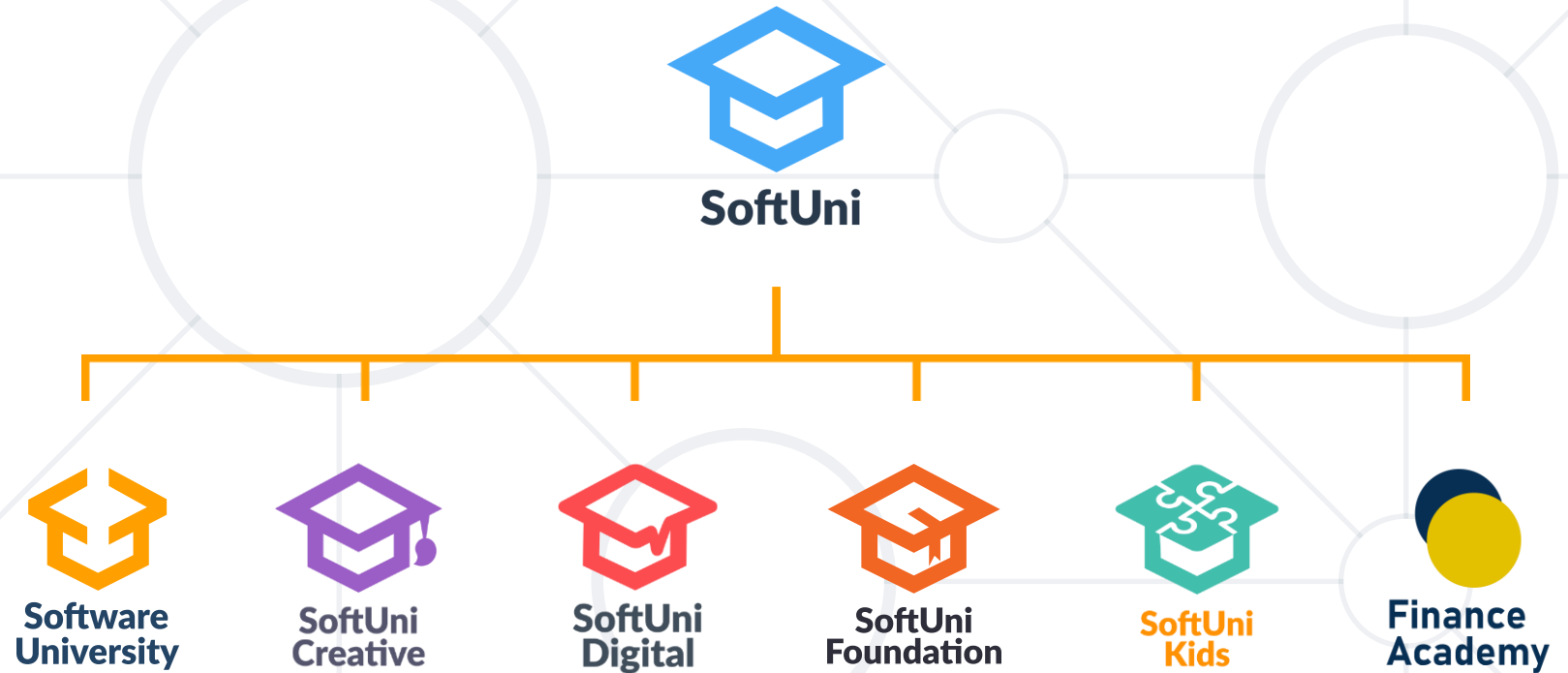
Live Exercise in Class (Lab)



- Stack
  - **LIFO** data structure
- Queue
  - **FIFO** data structure
- Stack and Queue in Python



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**

 **Flutter**<sup>TM</sup>  
International

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**BOSCH**

 **Postbank**  
*Решения за твоето утре*

 **PHAR  
VISION**



**SmartIT**

**DXC**  
TECHNOLOGY

createX  


- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction, or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

