Day 03

Recap

- Bubble Sort
- Linear Search
- Binary Search

Stack

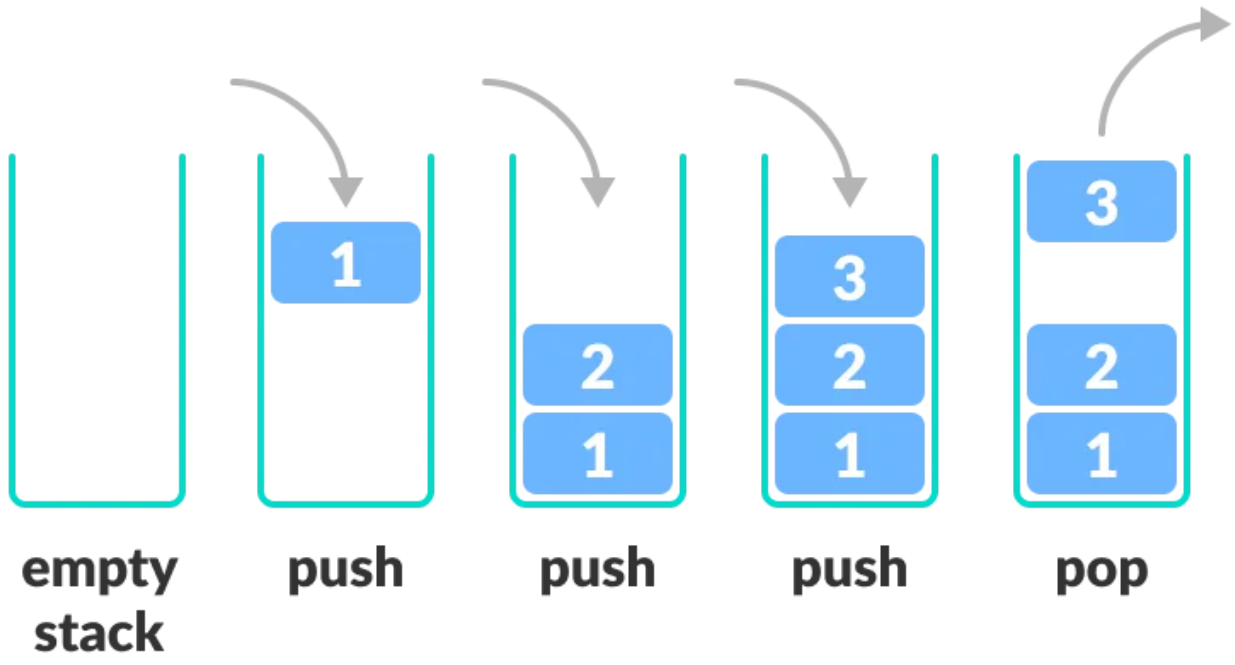
Property - LIFO -> Last In First Out

Queue

Property - FIFO -> First In First Out

Stack

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.



- LIFO
- FIFO

Application of Stack in real life:

- Undo and Redo mechanism in text editors.
- Web browser page changes

Basic Operations of Stack

There are some basic operations that allow us to perform different actions on a stack.

- **Push**: Add an element to the top of a stack
- **Pop**: Remove an element from the top of a stack
- **IsEmpty**: Check if the stack is empty
- **IsFull**: Check if the stack is full (arrays)
- **Peek**: Get the value of the top element without removing it

ATM PIN

4 Digits, Integer

Gmail Password

No resistriction, Alpha, Number, Special (Int/str)

- C language array group of data len/ data type
- Python list - group of data (mixed of data types)/no length

int arr[5] = 0

[1, 2, 3, 4, 5]

li = []

Stack

- List/ Arrays
- Linked List

In [14]:

```
class Stack:
    def __init__(self):
        self.stack = []
#     inserting data into the stack
    def push(self, data):
        self.stack.append(data)

#     to view the last eleent of the stack
    def peek(self):
        return self.stack[-1]

#     to remove last element from the stack
    def pop(self):
#         check stack is not empty then remove last element from stack
        if not self.isempty(): # Empty False True
            data = self.stack[-1]
            del self.stack[-1]
            return data
        else:
            return None
#   to check stack is empty or not
    def isempty(self):
        return self.stack == []
#     [] == [] -> true
#     [1] == [] -> false
```

In [15]:

```
s = Stack()
```

In [16]:

```
s.push(1)
s.push(2)
s.push(3)
```

In [17]:

```
s.peek()
```

Out[17]:

```
3
```

In [18]:

```
s.pop()
s.pop()
s.pop()
```

Out[18]:

```
1
```

In [20]:

```
1 print(s.pop())
```

None

In [11]:

```
1 s.pop()
```

Out[11]:

2

In [21]:

```
1 s.isempty()
```

Out[21]:

True

In [13]:

```
1 s.pop()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-13-c88c8c48122b> in <module>
----> 1 s.pop()

<ipython-input-5-80613f867c49> in pop(self)
     11
     12     def pop(self):
---> 13         data = self.stack[-1]
     14         del self.stack[-1]
     15         return data

IndexError: list index out of range
```

In [22]:

```
1 s.isempty()
```

Out[22]:

True

In [23]:

```
1 s.push(1)
```

In [24]:

```
1 s.isempty()
```

Out[24]:

```
False
```

In [25]:

```
1 s.peek()
```

Out[25]:

```
1
```

Stack Memory

small amount of special memory RAM

In [29]:

```
1 def f1():
2     d = 1
3     e = 2
4     return f2(d, e)
5 def f2(a, b):
6     return f3(a+b, a-b)
7 def f3(b, c):
8     return b * c
```

f1 -> f2 -> f3

f1 -> f2

f1

In [30]:

```
1 f1()
```

Out[30]:

```
-3
```

In [31]:

```
1 print(d)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-85549cb1de5f> in <module>
----> 1 print(d)

NameError: name 'd' is not defined
```

Queues

- it is an abstract data type – and it can be implemented either with arrays or with linked lists
- it has a so-called FIFO/LILO structure – the first item we inserted is the first item we take out

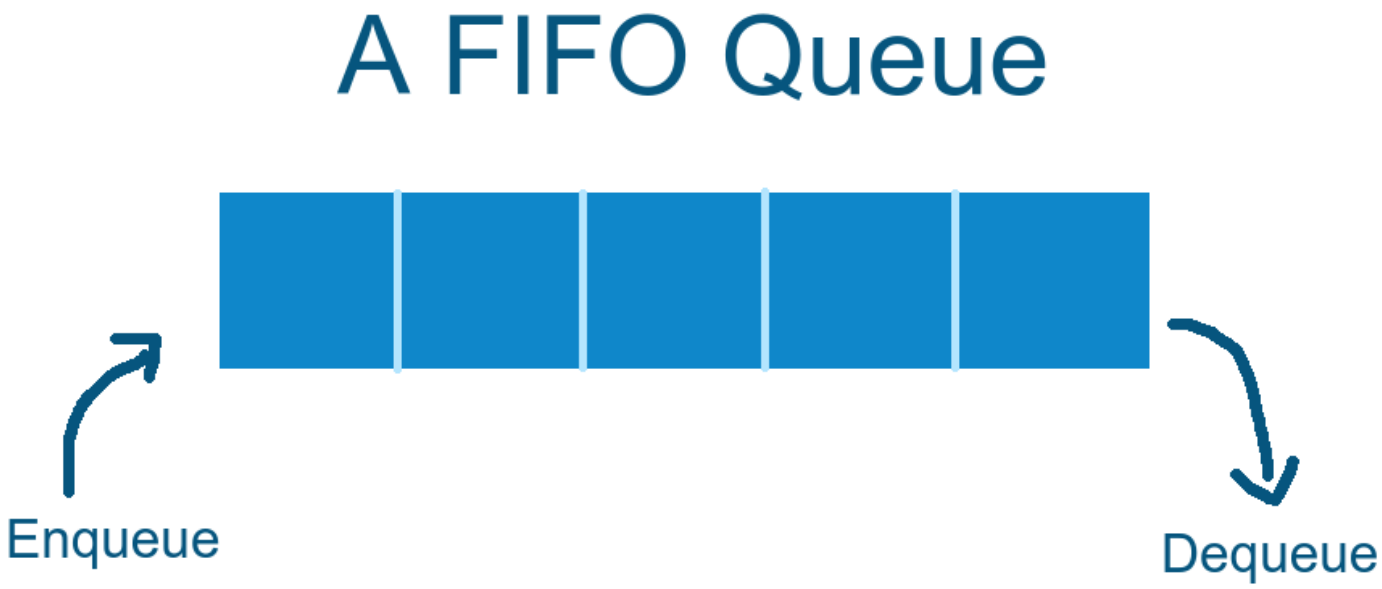
Basic operations are

- **enqueue()**: Adding to the queue (at the tail or end)
- **dequeue()**: Removing from the queue (at the start or head)
- **peek()**: Shows the data of top element

li = [1,2,3]

li = [1,2,3,4] -> tail -> 4

li = [1,2,3,4] -> head -> 1



- CPU Scheduling
  - increasing volume
  - slient mode
  - aeroplane mode
  - power button
- Bank/ Ticket Counter
- waiting list 3 conformed

5 tick, 3 tick, 2 tick -> 3 ticket cancled to FIFO

In [33]:

```
class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        data = self.queue[0]
        del self.queue[0]
        return data

#     what is the element added first or enqueued
    def peek(self):
        return self.queue[0]

    def size(self):



    def isempty(self):

```

In [34]:

```
qu = Queue()
```

In [35]:

```
qu.enqueue(1)
```

In [36]:

```
qu.peek()
```

Out[36]:

1

In [37]:

```
qu.enqueue(2)
qu.enqueue(3)
qu.enqueue(4)
qu.enqueue(5)
```

In [38]:

```
qu.peek()
```

Out[38]:

1

In [39]:

```
qu.dequeue()
```

Out[39]:

1

In [40]:

```
qu.dequeue()
```

Out[40]:

2

In [41]:

```
qu.peek()
```

Out[41]:

3

In [42]:

```
1 qu.enqueue(1)
2 qu.enqueue(2)
```

In [43]:

```
1 qu.peek()
```

Out[43]:

3

- When queue empty
  - peek
  - deqeue

In [ ]:

```
1
```