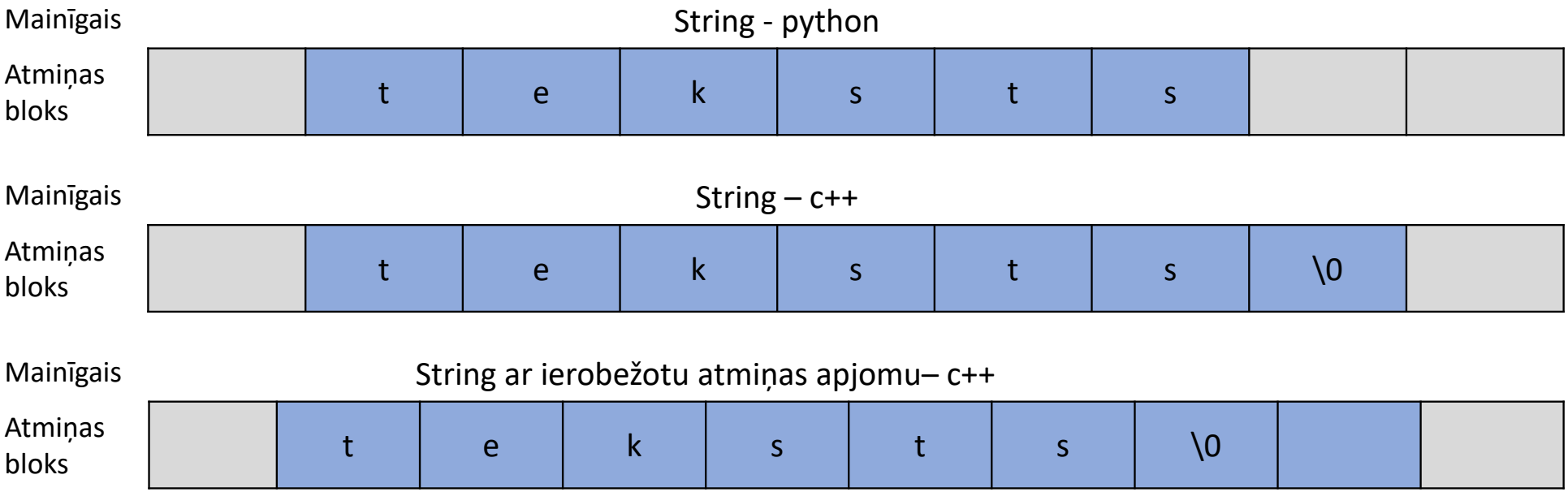
STACK and QUEUE

Datu tipu klasifikācija

- Skalārs datu tips
  (sastāv no vienas vērtības)
  - Predefinēts – bāzes tipi
  - Lietotāja definēts – diapazona, rādītāja, references, uzskaitāmais
- Strukturēts datu tips
  (sastāv no vairākām vērtībām)
  - Predefinēts – datnes, ieraksti, kopas, masīvi, virknes
  - Lietotāja definēts – grafi, koki, rindas, saraksti, steki, tabulas

Datu tipa attēlojums teksta virknēm



Simbolu virknes var būt ar:

- Neierobežotu garumu (dinamiski mainīgu)
- Fiksētu garumu (vienāds ar ievadīto garumu)
- Ierobežotu garumu (nevar pārsniegt norādīto)

Datu tipa attēlojums struktūrām

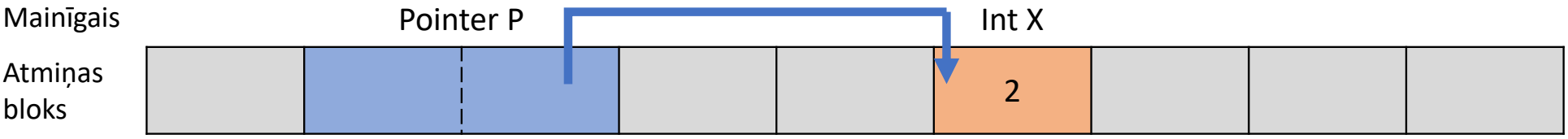
Ieraksts satur vairākus dažādu datu tipu mainīgos

Piemēram:

- Studenta ID
- Datums
- Vērtējums

		Studenta vērtējums					
Mainīgais		Studenta ID	Datums	Vērtējums			
Atmiņas bloks		15	30.12.2022.	9			

Datu tipu attēlojums norādēm un masīviem



Mainīgais: Array

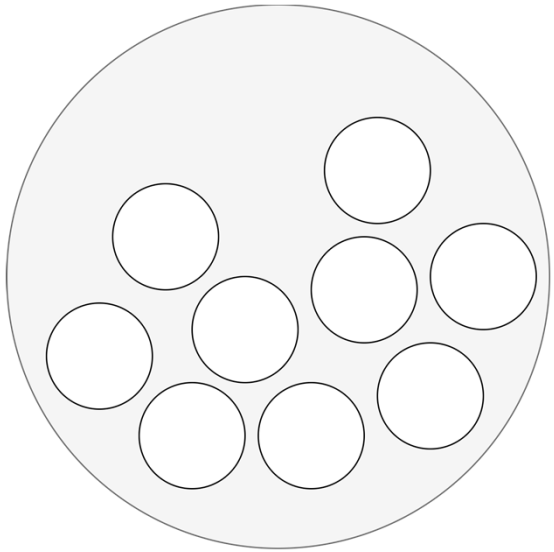
Atmiņas bloks		1	2	3	5	4	8		
Indeks		0	1	2	3	4	5		
Vieta atmiņā	201	202	203	204	205	206	207	208	209

DS elementu saites – nav saites

DS, kurā starp elementiem nav saišu, kuri pieder noteiktam datu kopumam (mainīgajam), bet nav pirmā/pēdējā elementa, sauc par kopu.

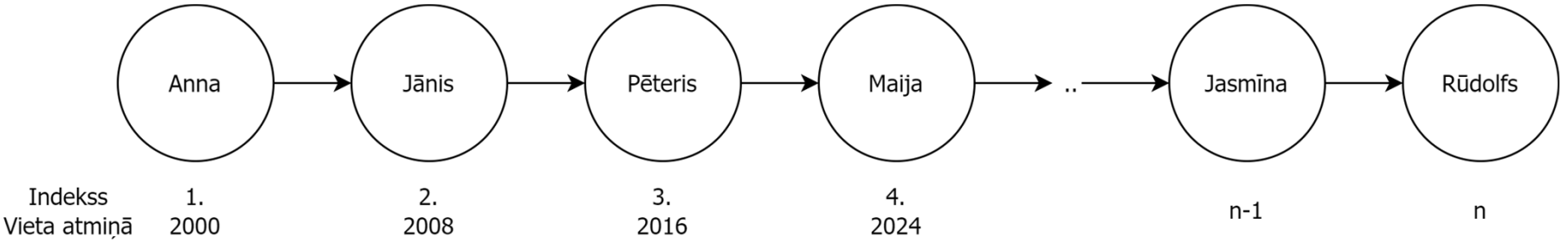
- Saite – relationship
- Kopa – Set

Piemēram, veikala pircēji



DS elementu saites – viens ar vienu - pozicionēšana

Elementi tiek izvietoti atmiņā viens aiz otra jeb secīgi



Ja ir zināma i-tā saraksta elementa adrese,
tad i+m-tā elementa adrese ir aprēķināma kā

$$adrese_{i+1} = adrese_i + m$$

m – lauka garums m = 8 (elem.garums)
i – kārtas nr i = 1,2,.., n-1

Šo paņēmienu sauc par **pozicionēšana** (positioning, array represantation)

DS elementu saites – viens ar vienu – linked lists

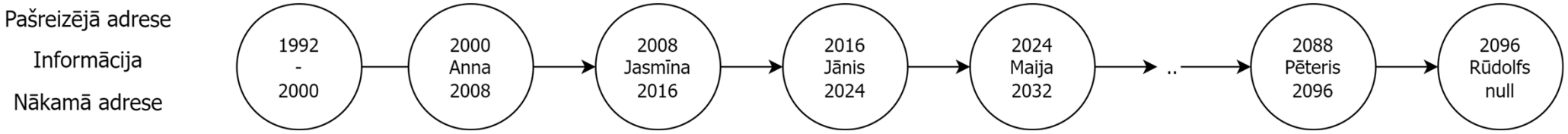
Pirmā elementa adrese glabājas speciālā rādītājā (pointer)

Katrs elements satur arī nākamā elementa adresi

Pēdējam elementam nav pēcteča – glabā nil (null)

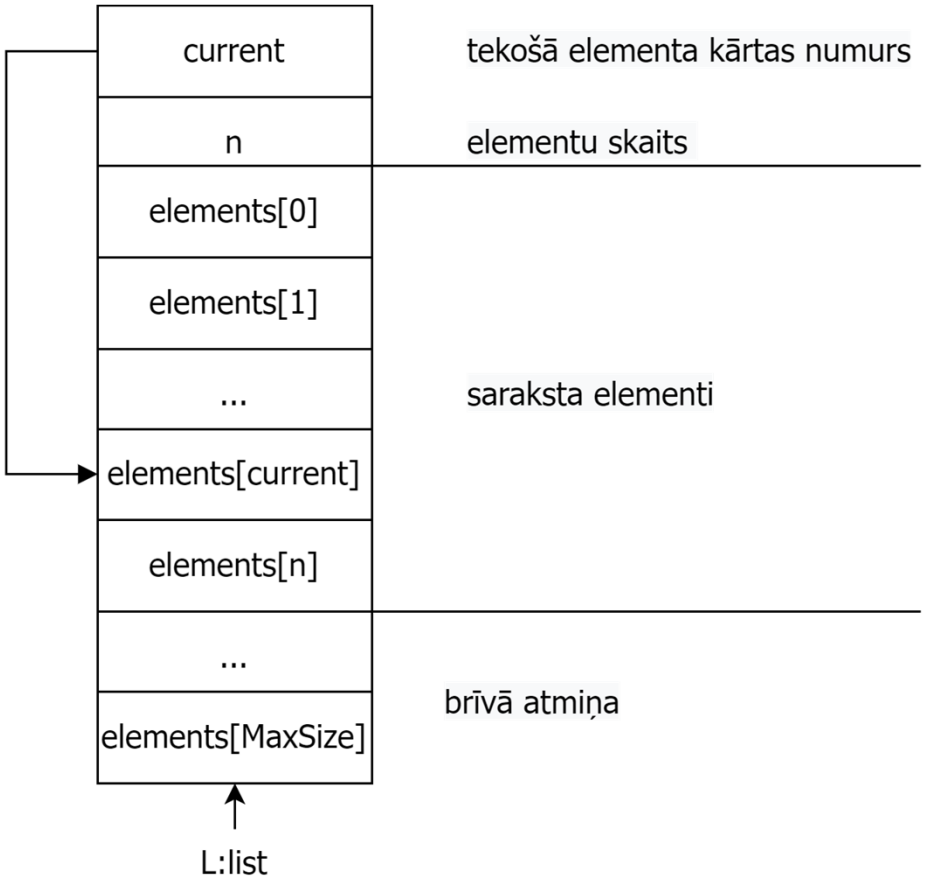
Lai nodrošinātu piekļuvi jebkuram saraksta elementam, nepieciešams apstrādājamā jeb tekošā elementa rādītājs.

Šo paņēmienu sauc par **saistīšanu** (linking, linked represantation)



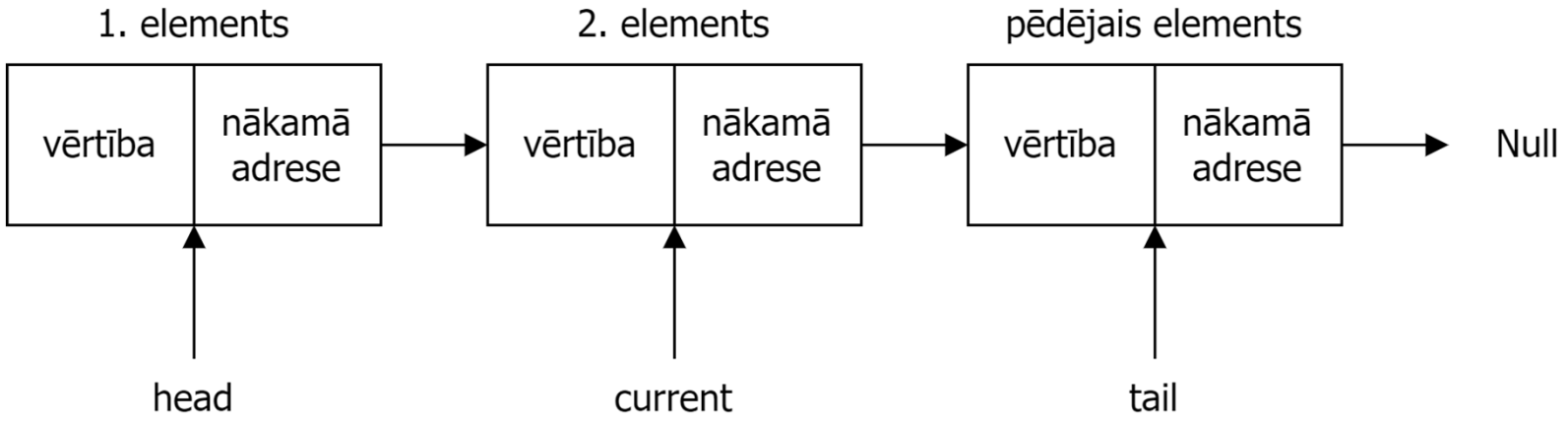
DS attēlošana – vektoriāla forma

Pozicionēšana



DS attēlošana – saistītā forma

Saistīšana



Paškontroles jautājums

Vērtība	1	5	17	3	25	32	14	21
Indekss	0							
adrese	1000							

Masīvs sākas ar adresi 1000

Masīva elementa izmērs ir 8 baiti

Pirmais indekss ir 0

Kāda ir adrese 6-tajā indeksā?

Paškontroles jautājums

Vērtība	1	5	17	3	25	32	14	21
Indekss	0							
adrese	1000							

Masīvs sākas ar adresi 1000

Masīva elementa izmērs ir 8 baiti

Pirmais indekss ir 0

Kāda ir adrese 6-tajā indeksā?

1040 / 1005 / 1048 / 1006 / 40 / 48

Paškontroles jautājums

Vērtība	1	5	17	3	25	32	14	21
Indekss	0	1	2	3	4	5	6	7
adrese	1000							

Masīvs sākas ar adresi 1000

Masīva elementa izmērs ir 8 baiti

Pirmais indekss ir 0

Kāda ir adrese 6-tajā indeksā?

1040 / 1005 / 1048 / 1006 / 40 / 48

Paškontroles jautājums

Vērtība	1	5	17	3	25	32	14	21
Indekss	0	1	2	3	4	5	6	7
adrese	1000	1008	1016	1024	1032	1040	1048	1056

Masīvs sākas ar adresi 1000

Masīva elementa izmērs ir 8 baiti

Pirmais indekss ir 0

Kāda ir adrese 6-tajā indeksā?

1040 / 1005 / 1048 / 1006 / 40 / 48

Paškontroles jautājums

Single linked list satur: 10, 11, 12, 13

Tiek izpildīts kods:

```
p <- head
while p.next.next != NULL:
    p <- p.next
```

Cik reizes izpildīsies kods, ja sākumā p norāda uz 10?

Paškontroles jautājums

Single linked list satur: 10, 11, 12, 13

Tiek izpildīts pseido kods:

```
p <- head
while p.next.next != NULL:
    p <- p.next
```

Cik reizes izpildīsies kods, ja sākumā p norāda uz 10?

10.next.next = 12
11.next.next = 13
12.next.next = NULL

Datu struktūra – rinda (queue)

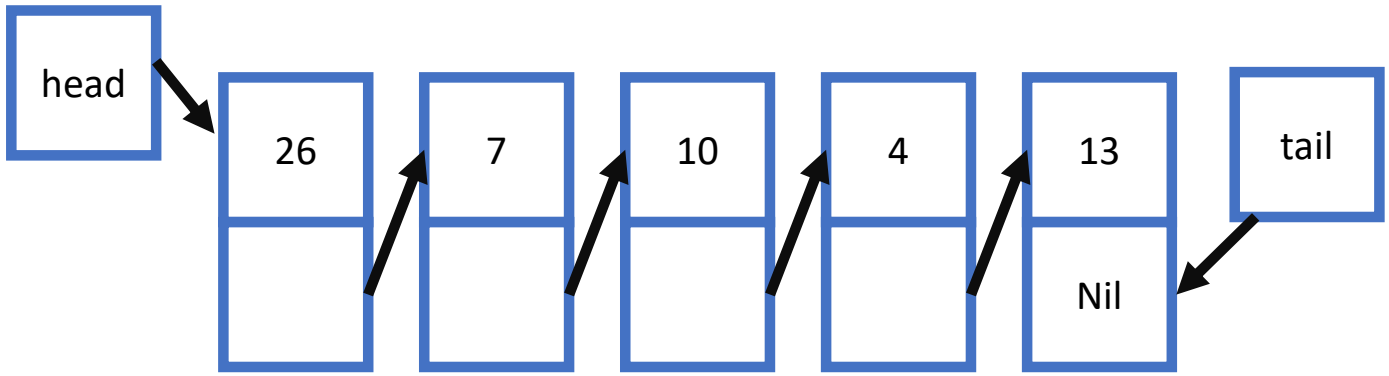
Rinda ir lineāra datu struktūra, kas strādā pēc principa:

- FIFO – First in, first out (pirmais iekša, pirmais ārā)
- HPIFO – High priority in, first out (augstas prioritātes iekšā, pirmais ārā) – prioritārā rinda

Īpašibas:

- Var nepārtraukti papildināt ar jauniem elementiem
- Jaunie elementi pievienojas rindas beigās (prioritārajā rindā atrod vietu pēc prioritātes)
- Nolasa elementu no rindas sākuma
- Pēc nolasīšanas elements tiek dzēsts

Rinda



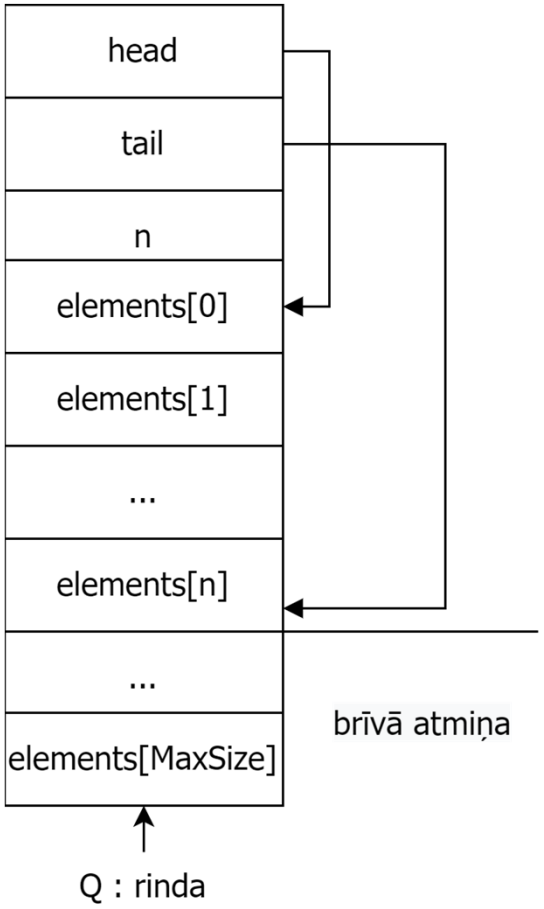
Datu struktūra – rinda (queue)

Rādītāji

- Head – norāde uz rindas sākumu
- Tail – norāde uz rindas pēdējo elementu
- Visiem izņemot pēdējo elementu ir pēctecis.

Apstrādes operācijas

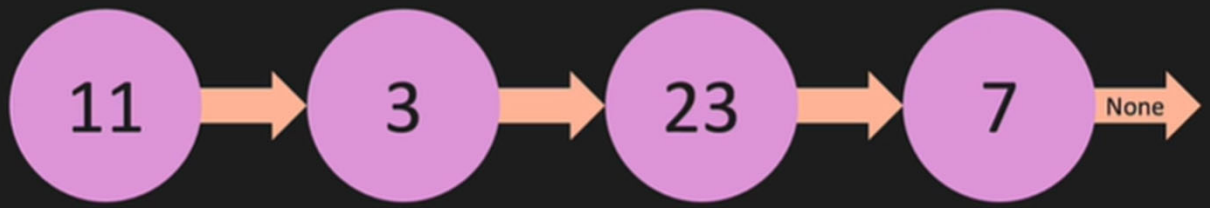
- Apkalpošanas operācijas
  - Enqueue/ Insert (push, append)
  - Dequeue / Remove / Serve (pop)
- Pamatoperācijas
  - Create
  - Size
  - Empty
  - Full



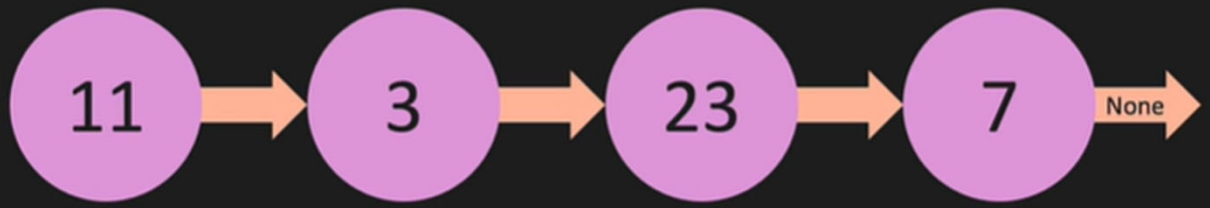
Rinda - Queue

Enque(Key) – pievieno vērtību List.PushBack	O(1)
Key Dequeue() – atgriež vērtību un dzēš vērtību	O(n), ja nav norāde uz tail
List.TopFront un List.PopFront	O(1), ja ir norāde tail
Empty() – pārbauda vai ir tukšs	O(1)

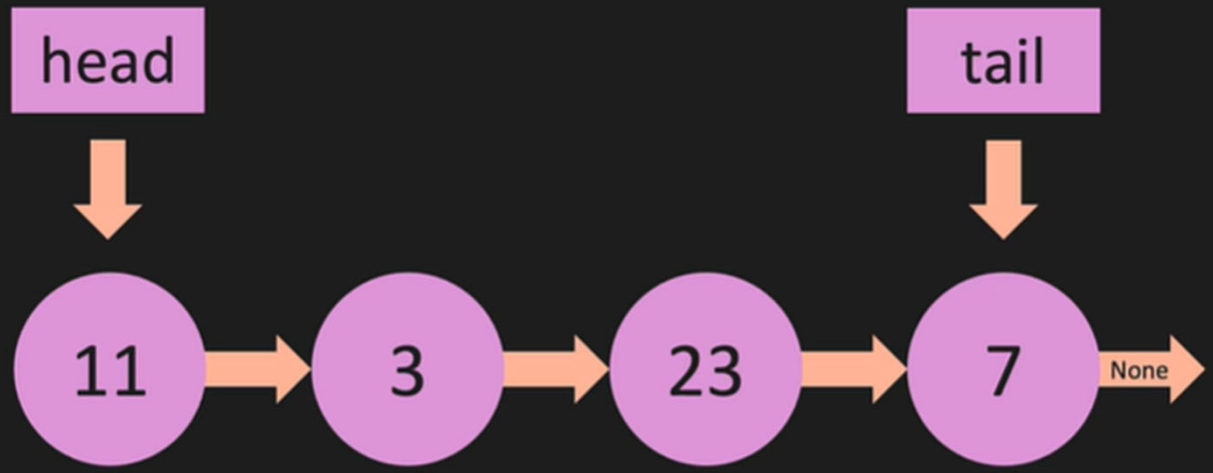

11
3
23
7
None



O(1) O(n)

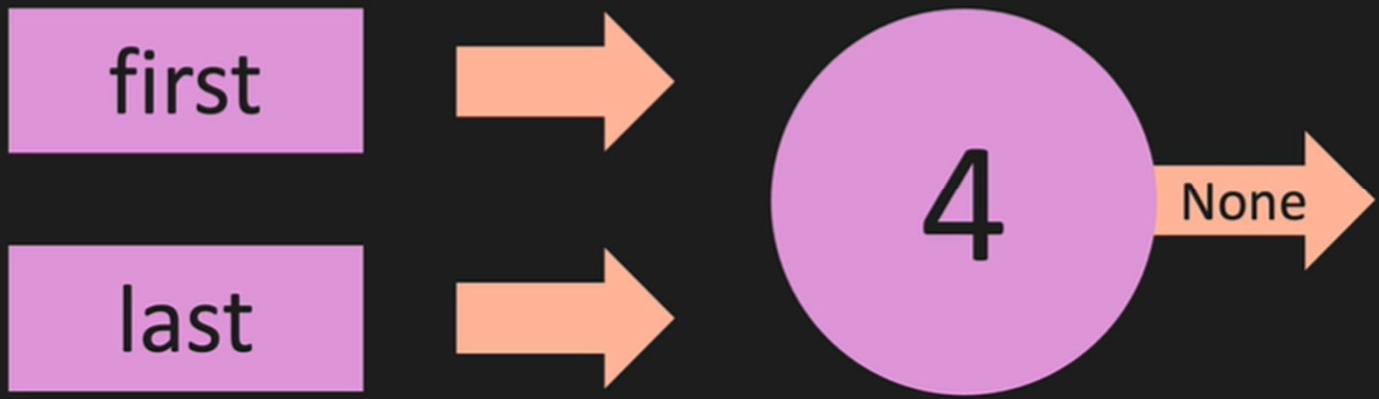


O(1) O(1) O(1) O(n)


head
tail
11
3
23
7
None

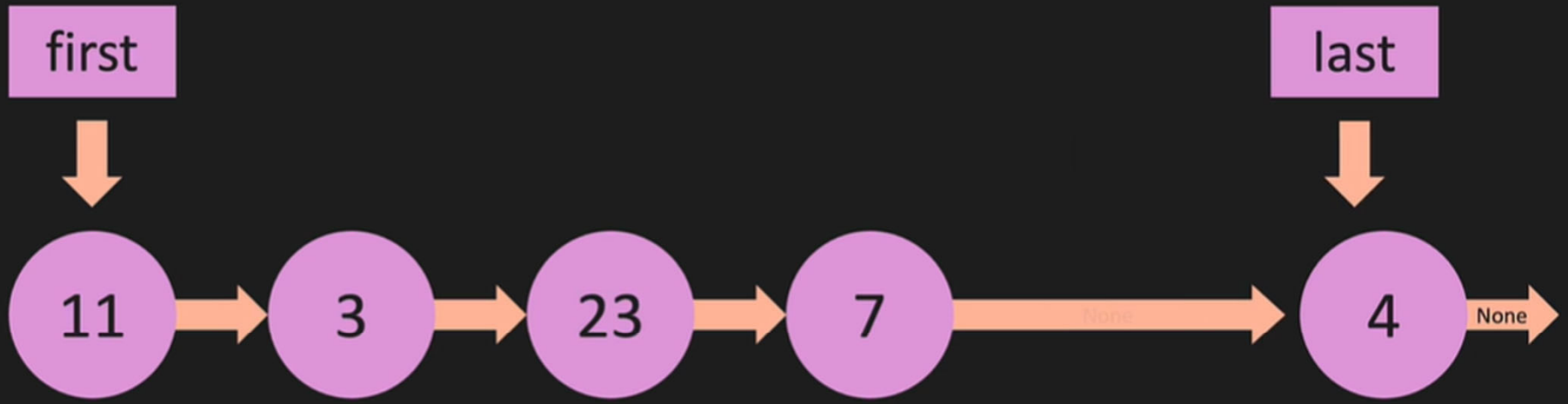
```python
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None
```

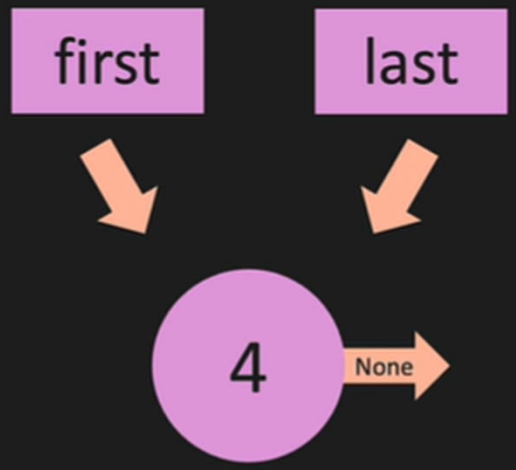
```
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
```



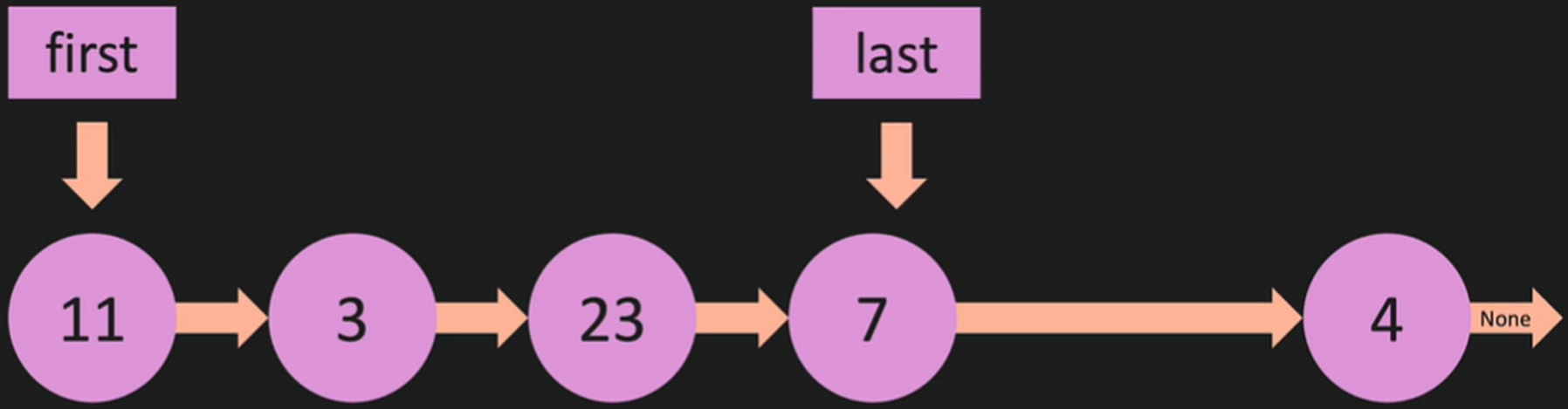
```
class Queue:
    def __init__(self, value):
        new_node = Node(value)
        self.first = new_node
        self.last = new_node
        self.length = 1


my_queue = Queue(4)
```

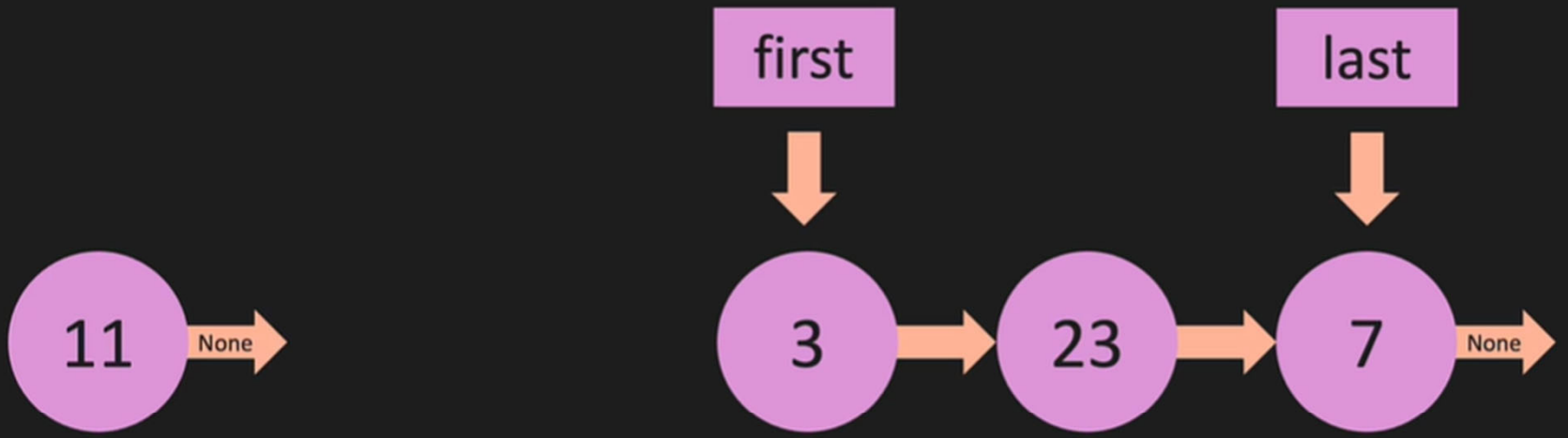

first
last
11
3
23
7
None
4
None



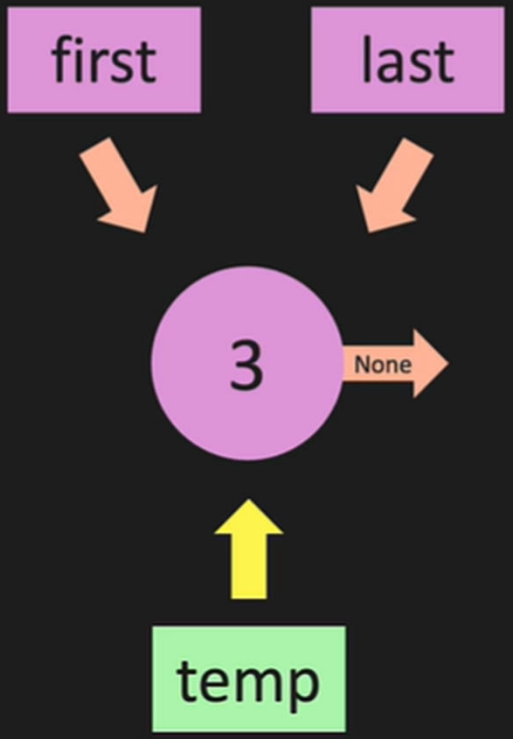

```
if self.first is None:
    self.first = new_node
    self.last = new_node
```



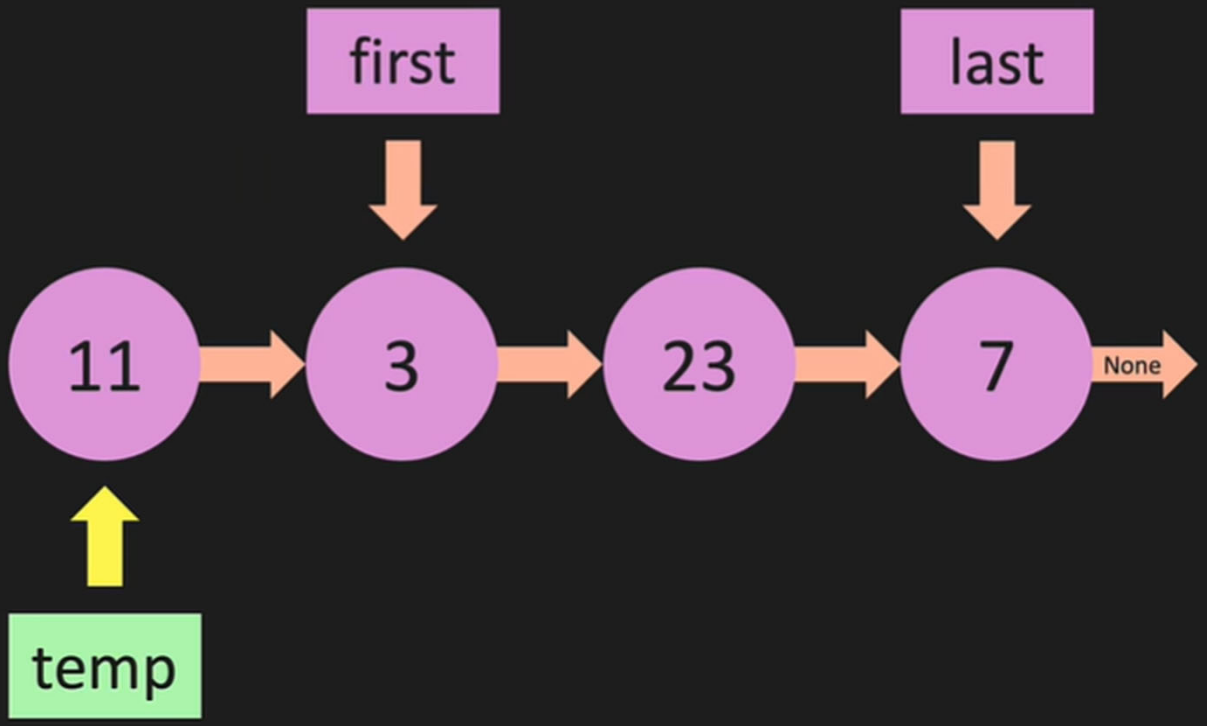

```
else:
    self.last.next = new_node
    self.last = new_node
```


first
last
11
None
3
23
7
None

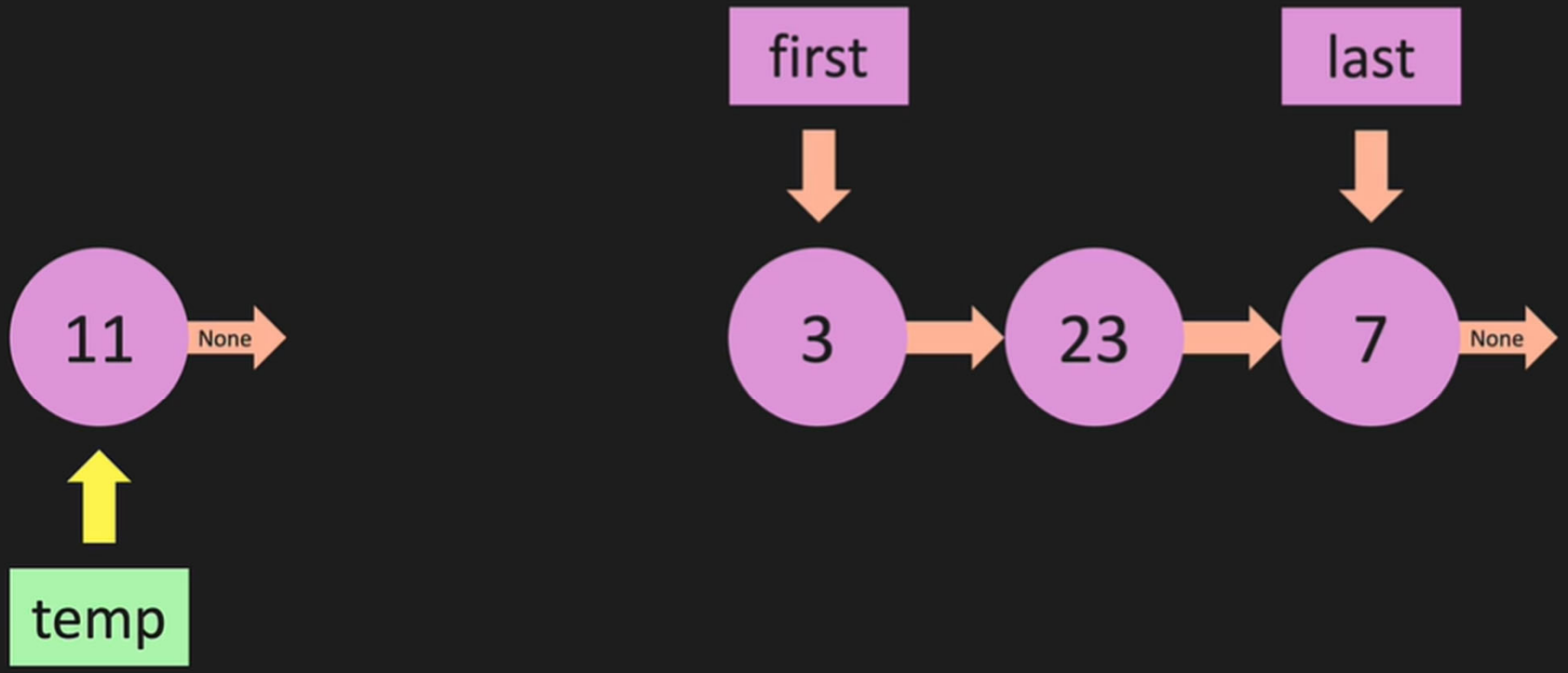
```
temp = self.first
if self.length == 1:
    self.first = None
    self.last = None
```



```
else:
    self.first = self.first.next
```



```
        else:
            self.first = self.first.next
            temp.next = None
```

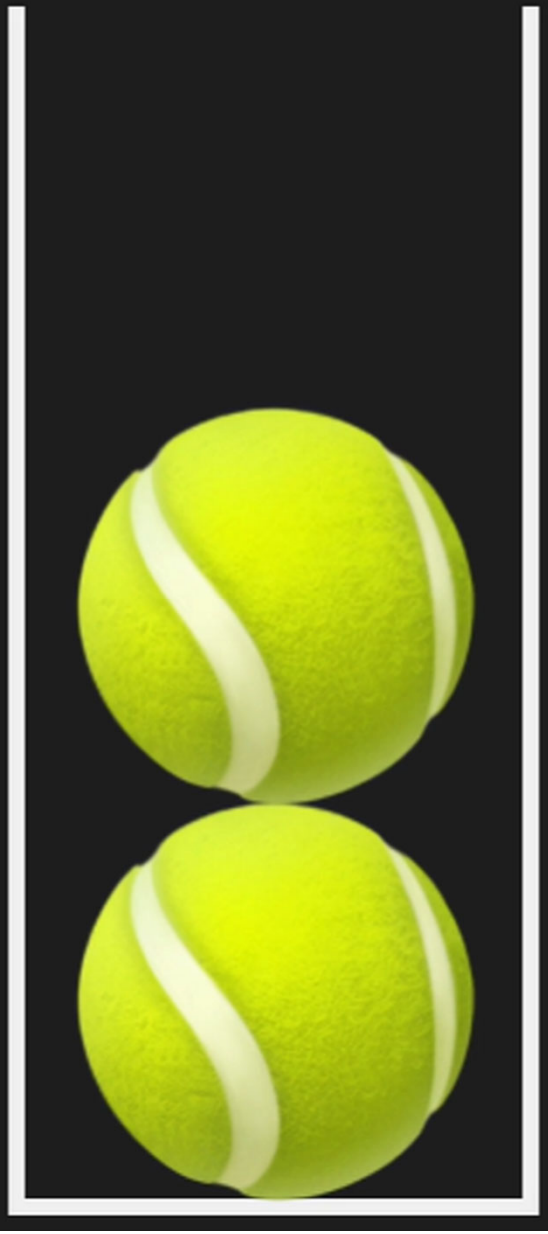


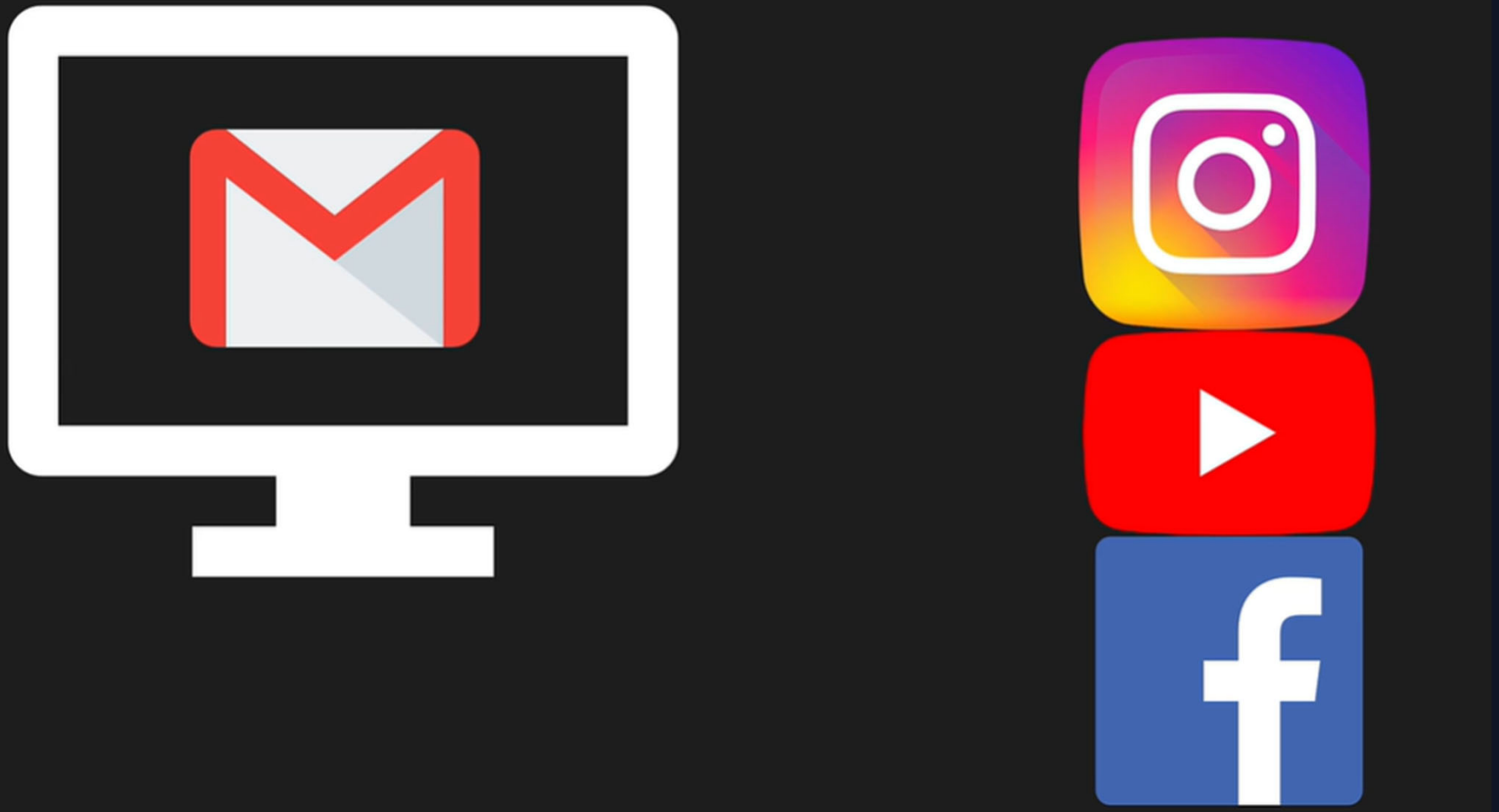
Steks

Steks ir abstrakts datu tips, kas darbojas pēc principa LIFO (last in, first out), kur pēdējā pievienotā vērtība ir pirmā pieejamā vērtība un nav iespējams piekļūt citām vērtībām.

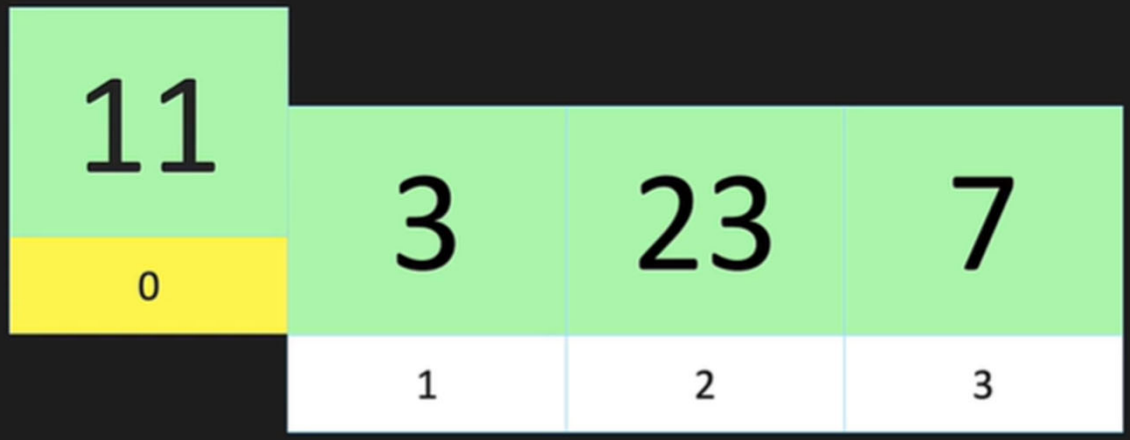
Starp elementiem ir saite.

Norāde tikai uz pēdējo pievienoto elementu.



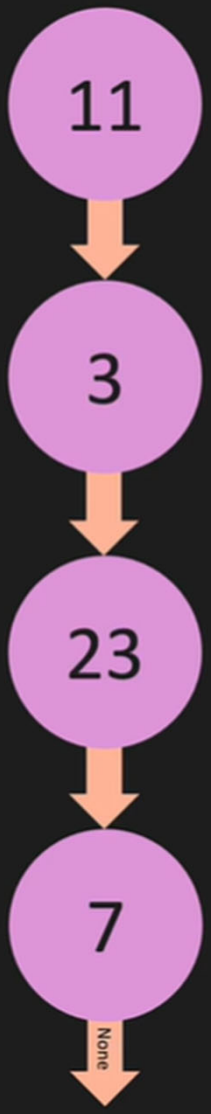


11	3	23	7
0	1	2	3



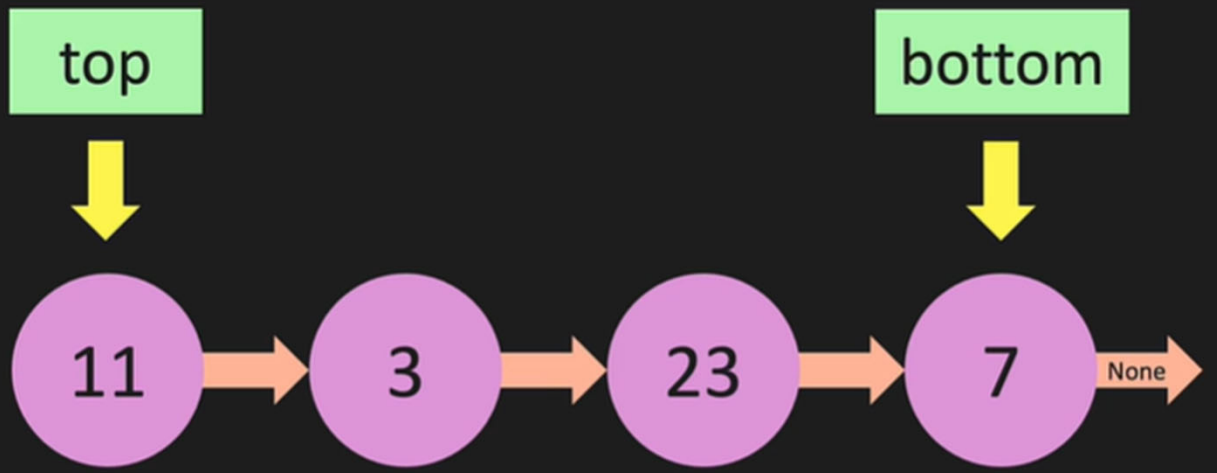
O(n)

11	3	23	7
0	1	2	3

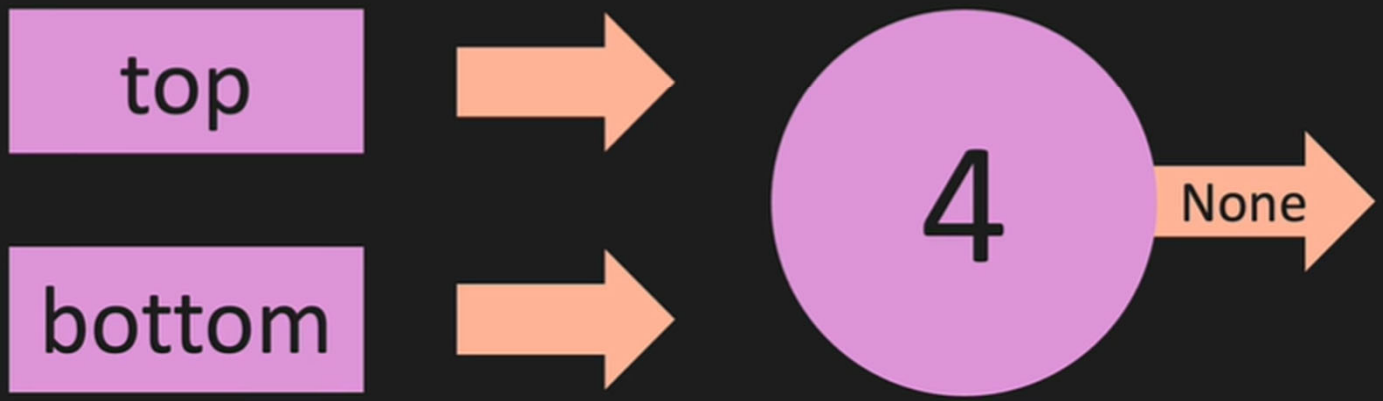


O(1)

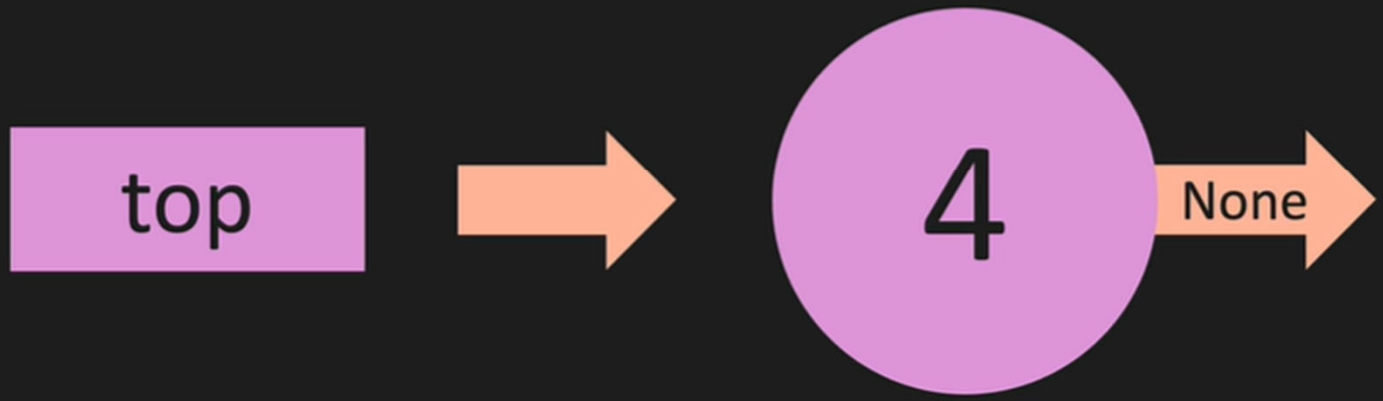
O(1)

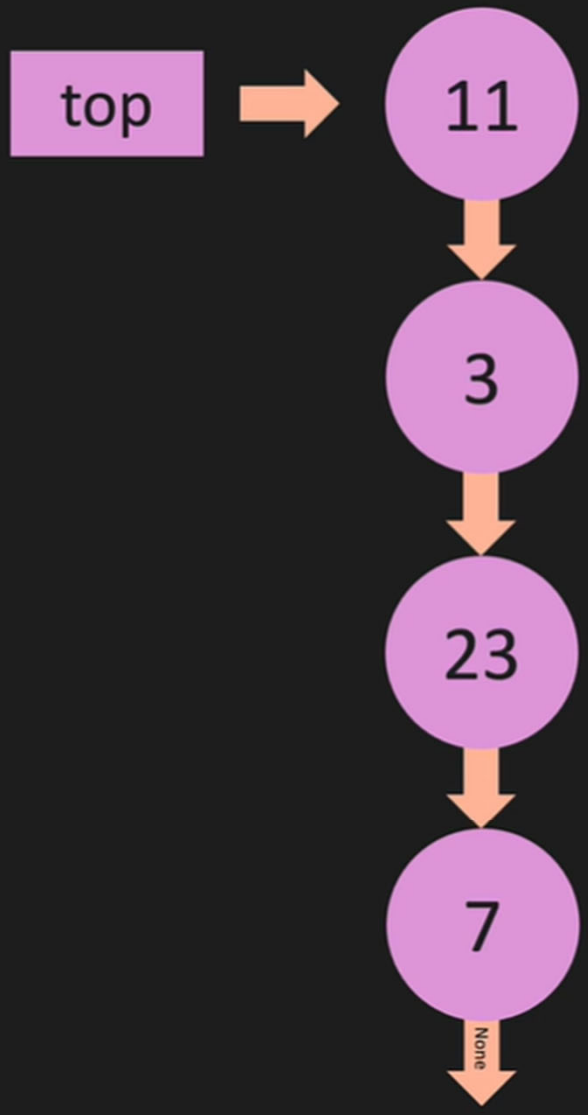

top
bottom
11
3
23
7
None

```
class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.bottom = new_node
```

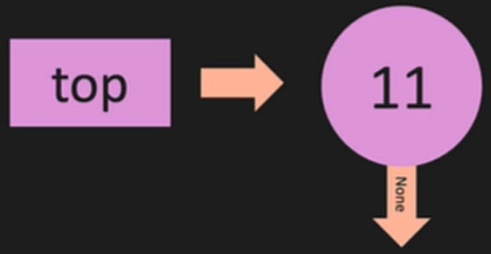


```
class Stack:
    def __init__(self, value):
        new_node = Node(value)
        self.top = new_node
        self.height = 1
```

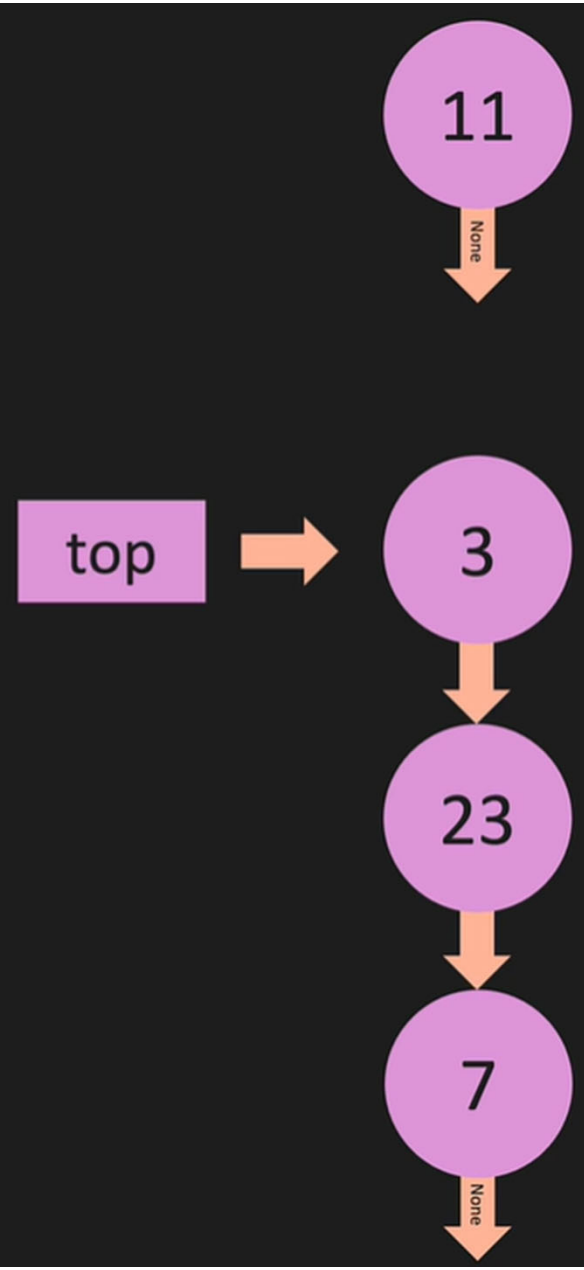




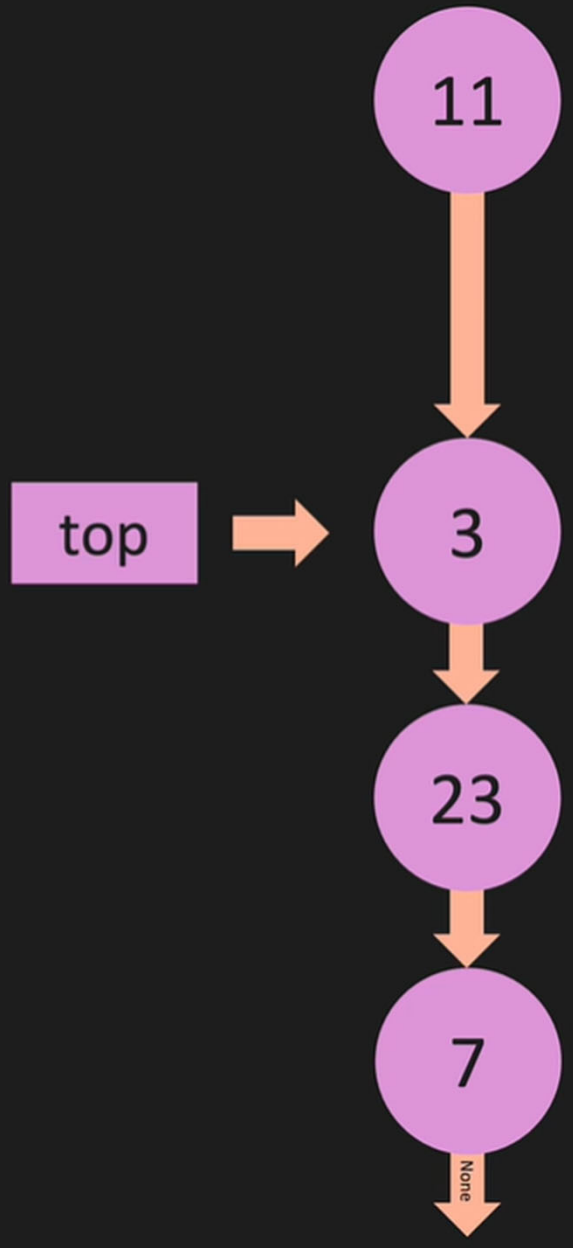

```
def push(self, value):
    new_node = Node(value)
    if self.height == 0:
        self.top = new_node
    else:
```

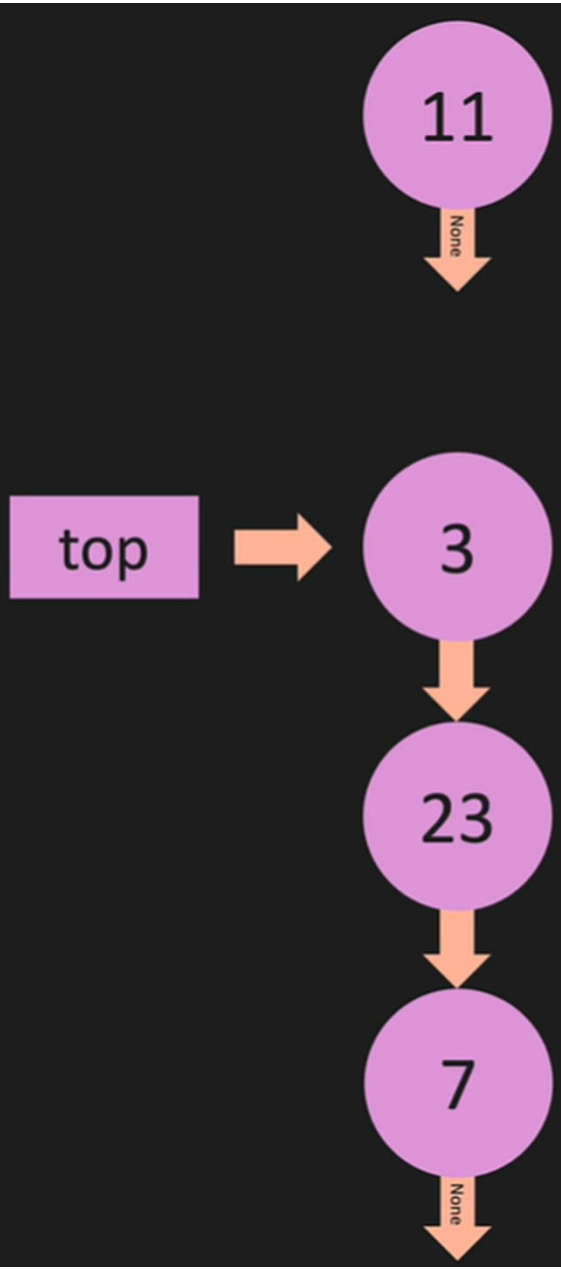


```
else:
```



```
else:
    new_node.next = self.top
    self.top = new_node
```






```
def pop(self):
    if self.height == 0:
        return None
```

```
temp = self.top
self.top = self.top.next
temp.next = None
```