Structures de données abstraites

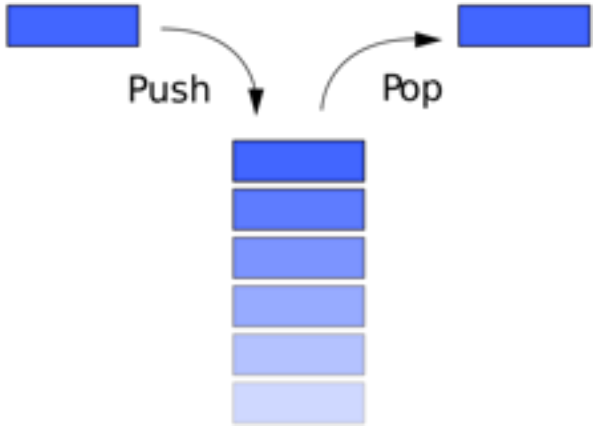
Piles (stack)

Principe

Interface (fonctionnalités offertes)

Piles (stack)

Principe

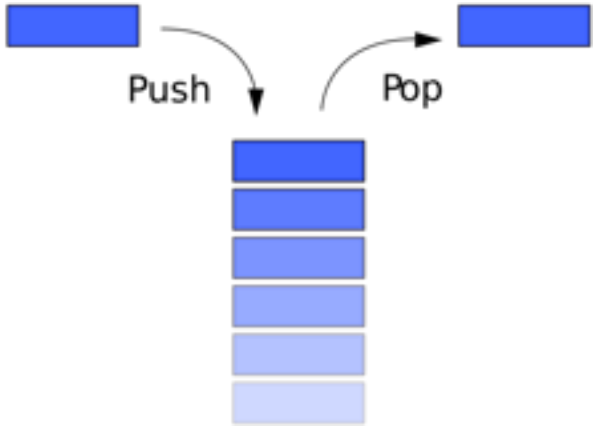


Interface (fonctionnalités offertes)

Piles (stack)

Principe

- Structure linéaire

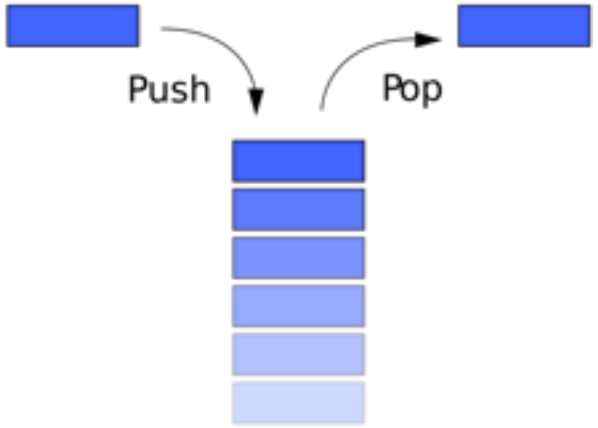


Interface (fonctionnalités offertes)

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)

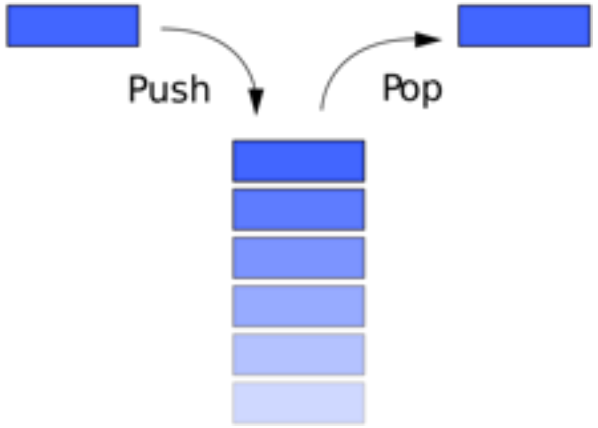


Interface (fonctionnalités offertes)

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)
- Exemple :
  - Pile de livres, d'assiettes
  - Bouton « précédent » de Firefox

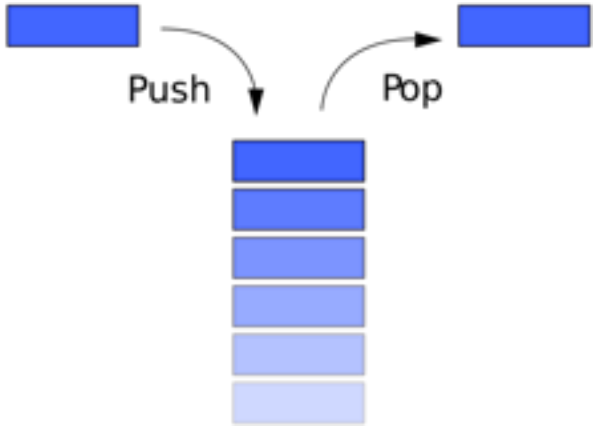


Interface (fonctionnalités offertes)

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)
- Exemple :
  - Pile de livres, d'assiettes
  - Bouton « précédent » de Firefox



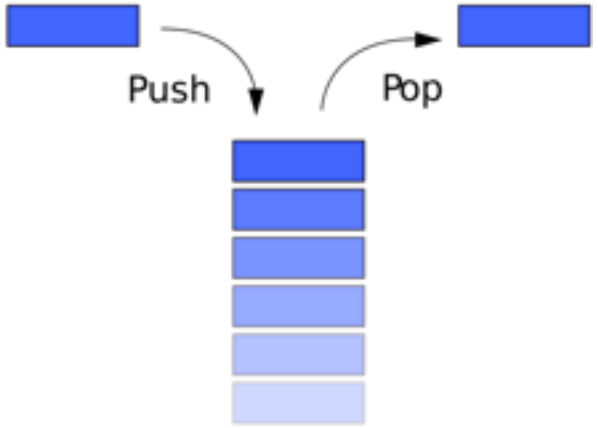
Interface (fonctionnalités offertes)

- **Stack**() : constructeur ➔ crée une pile

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)
- Exemple :
  - Pile de livres, d'assiettes
  - Bouton « précédent » de Firefox



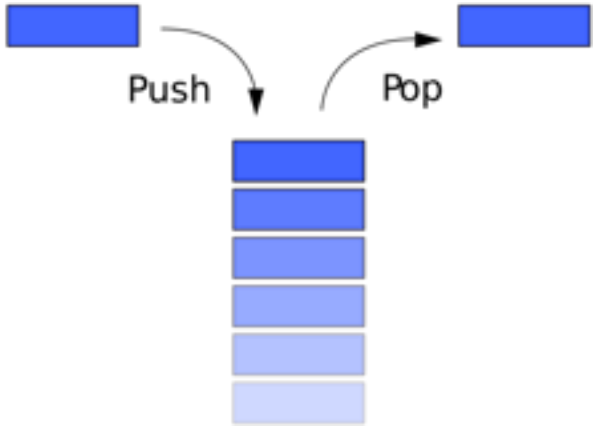
Interface (fonctionnalités offertes)

- **Stack**() : constructeur ➔ crée une pile
- **push(item)** : empile un item sur la pile

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)
- Exemple :
  - Pile de livres, d'assiettes
  - Bouton « précédent » de Firefox



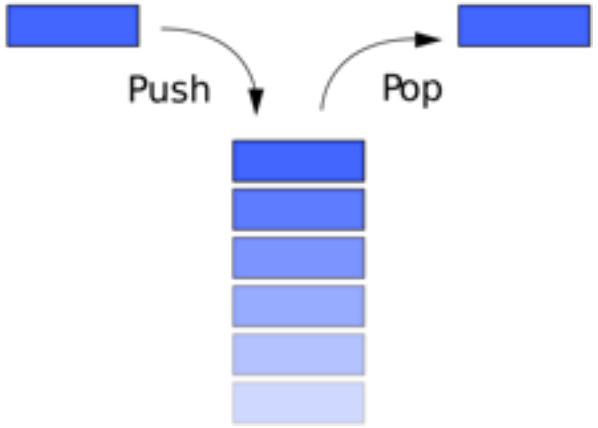
Interface (fonctionnalités offertes)

- **Stack**() : constructeur ➔ crée une pile
- **push(item)** : empile un item sur la pile
- **pop()** : enlève et retourne l'élément du haut de la pile

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)
- Exemple :
  - Pile de livres, d'assiettes
  - Bouton « précédent » de Firefox



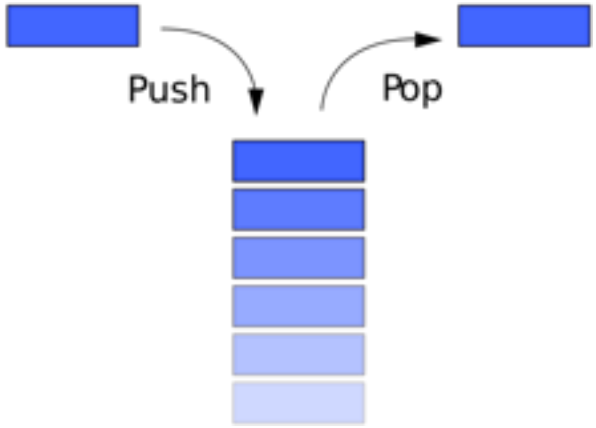
Interface (fonctionnalités offertes)

- **Stack**() : constructeur ➔ crée une pile
- **push(item)** : empile un item sur la pile
- **pop()** : enlève et retourne l'élément du haut de la pile
- **isEmpty()** : renvoie **True** si la pile est vide

Piles (stack)

Principe

- Structure linéaire
- Principe LIFO (Last In First Out)
- Exemple :
  - Pile de livres, d'assiettes
  - Bouton « précédent » de Firefox



Interface (fonctionnalités offertes)

- **Stack**() : constructeur ➔ crée une pile
- **push(item)** : empile un item sur la pile
- **pop()** : enlève et retourne l'élément du haut de la pile
- **isEmpty()** : renvoie **True** si la pile est vide
- **getSize() :** taille de la pile

Piles (utilisation)

Piles (utilisation)

```
s = Stack()
```

Piles (utilisation)

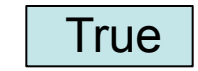
```
s = Stack()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
```



Stack

Piles (utilisation)

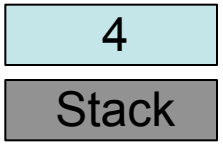
```
s = Stack()
s.isEmtpy()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
```

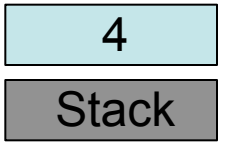
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
```



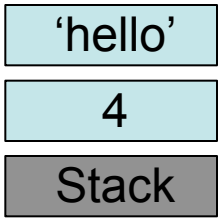
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
```



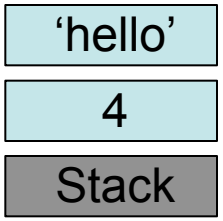
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
```



Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
```



Piles (utilisation)

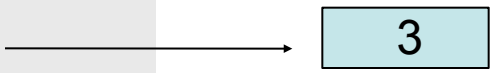
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
```



3

Piles (utilisation)

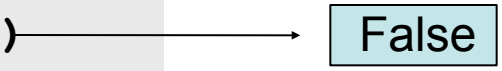
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
```



Piles (utilisation)

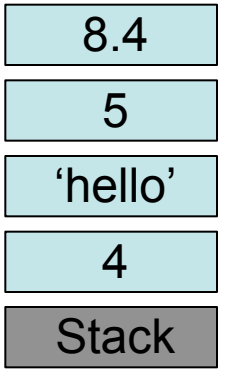
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
```

Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
```

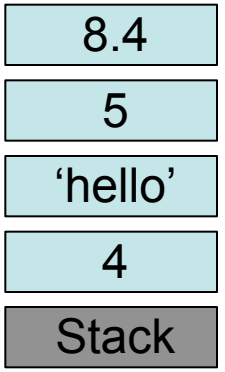
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
```



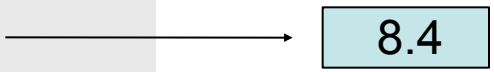
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
```



Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
```



Piles (utilisation)

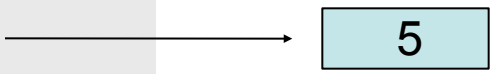
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
```

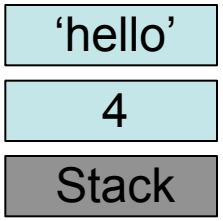
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
s.pop()
```

Piles (utilisation)

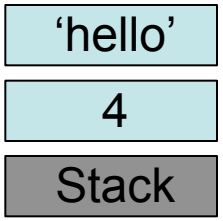
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
s.pop()
```





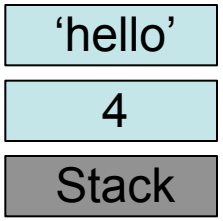
Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
s.pop()
```



Piles (utilisation)

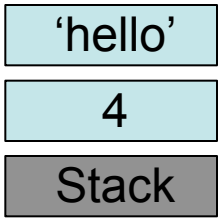
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
s.pop()
s.printContent()
```



Piles (utilisation)

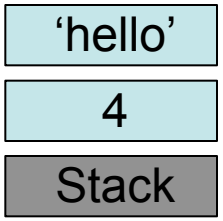
```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
s.pop()
s.printContent()
```

→ 'hello' 4



Piles (utilisation)

```
s = Stack()
s.isEmtpy()
s.push(4)
s.push('hello')
s.push(5)
s.size()
s.is_empty()
s.push(8.4)
s.pop()
s.pop()
s.printContent()
```



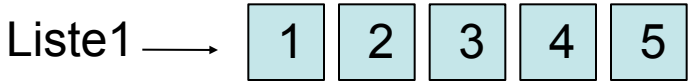
Application : inversion d’une liste

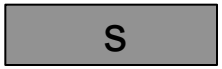
Application : inversion d'une liste

```
Liste1 = [1,2,3,4,5]
Liste2 = []
s = Stack()

for item in Liste1:
   s.push(item)

while not s.is_empty():
   Liste2.append(s.pop())
```





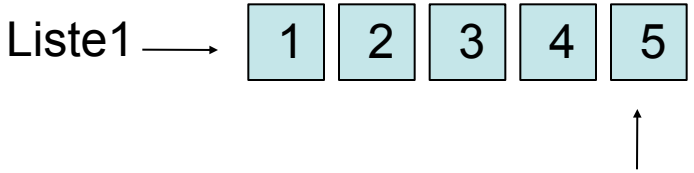
Liste2 ⟶

Application : inversion d'une liste

```
Liste1 = [1,2,3,4,5]
Liste2 = []
s = Stack()

for item in Liste1:
   s.push(item)

while not s.is_empty():
   Liste2.append(s.pop())
```



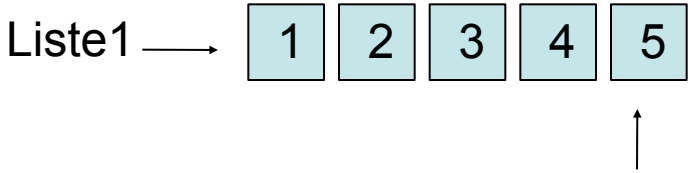
Liste2 ⟶

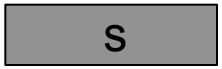
Application : inversion d'une liste

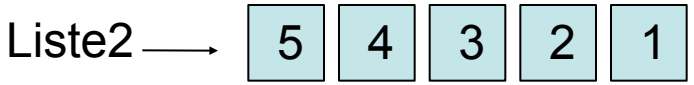
```
Liste1 = [1,2,3,4,5]
Liste2 = []
s = Stack()

for item in Liste1:
   s.push(item)

while not s.is_empty():
   Liste2.append(s.pop())
```







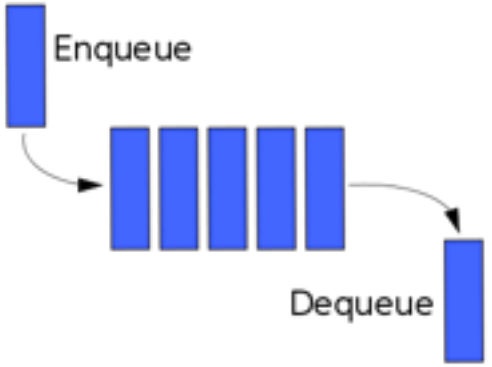
Files (Queues)

Principe

Interface (fonctionnalités offertes)

Files (Queues)

Principe

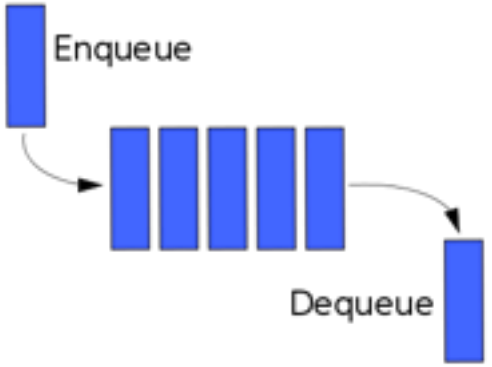


Interface (fonctionnalités offertes)

Files (Queues)

Principe

- Structure linéaire

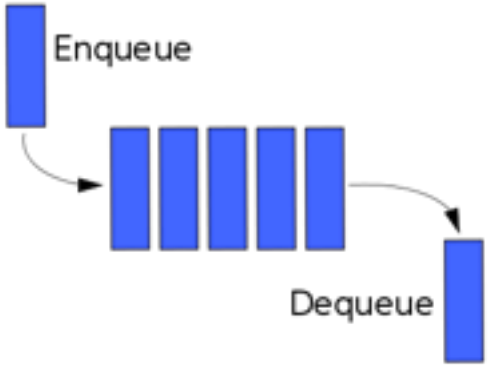


Interface (fonctionnalités offertes)

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)

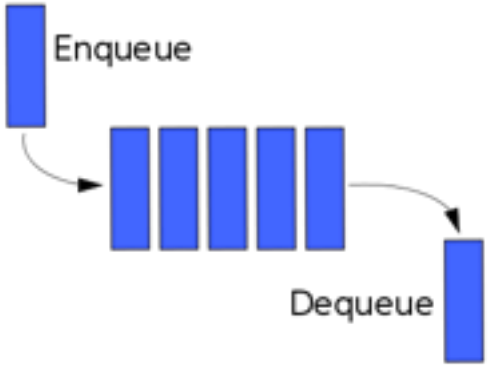


Interface (fonctionnalités offertes)

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)
- Exemple :
  - Queues à la cafétéria …
  - Files d'attente d'un serveur d'impression

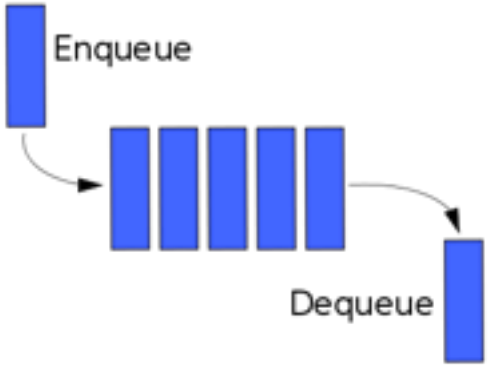


Interface (fonctionnalités offertes)

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)
- Exemple :
  - Queues à la cafétéria …
  - Files d'attente d'un serveur d'impression



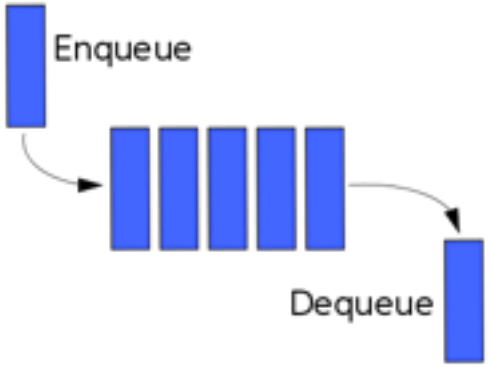
Interface (fonctionnalités offertes)

- **Queue**() : constructeur ➔ crée une file

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)
- Exemple :
  - Queues à la cafétéria …
  - Files d'attente d'un serveur d'impression



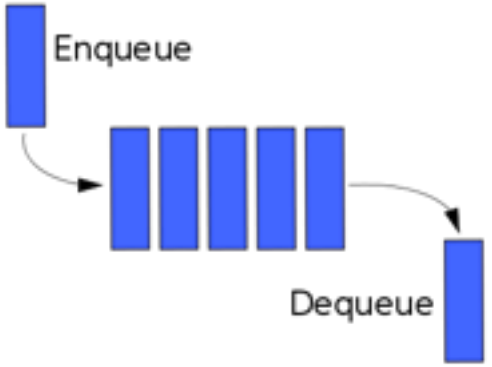
Interface (fonctionnalités offertes)

- **Queue**() : constructeur ➔ crée une file
- **enqueue(item)** : enfile un item vers l'arrière de la file

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)
- Exemple :
  - Queues à la cafétéria …
  - Files d'attente d'un serveur d'impression



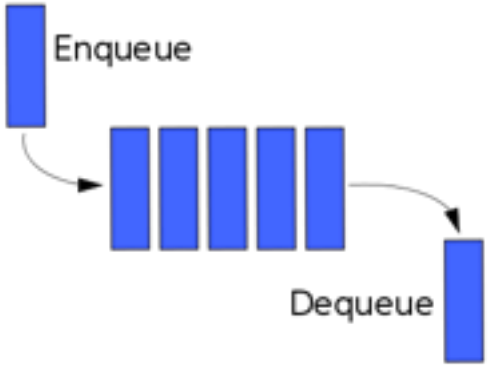
Interface (fonctionnalités offertes)

- **Queue**() : constructeur ➔ crée une file
- **enqueue(item)** : enfile un item vers l'arrière de la file
- **dequeue()** : enlève et retourne l'élément de l'avant de la file

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)
- Exemple :
  - Queues à la cafétéria …
  - Files d'attente d'un serveur d'impression



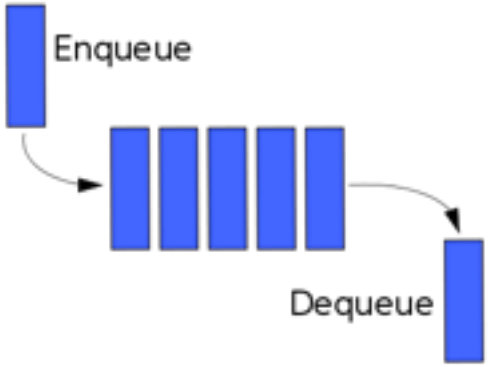
Interface (fonctionnalités offertes)

- **Queue**() : constructeur ➔ crée une file
- **enqueue(item)** : enfile un item vers l'arrière de la file
- **dequeue()** : enlève et retourne l'élément de l'avant de la file
- **isEmpty()** : renvoie **True** si la file est vide

Files (Queues)

Principe

- Structure linéaire
- Principe **FIFO** (First In First Out)
- Exemple :
  - Queues à la cafétéria …
  - Files d'attente d'un serveur d'impression



Interface (fonctionnalités offertes)

- **Queue**() : constructeur ➔ crée une file
- **enqueue(item)** : enfile un item vers l'arrière de la file
- **dequeue()** : enlève et retourne l'élément de l'avant de la file
- **isEmpty()** : renvoie **True** si la file est vide
- **getSize() :** taille de la pile

Files (utilisation)

Évolution de la file :

Files (utilisation)

```
q = Queue()
```

Évolution de la file :

q ⟶

Files (utilisation)

```
q = Queue()
q.is_emtpy()
```

Évolution de la file :

q ⟶

Files (utilisation)

```
q = Queue()
q.is_emtpy()
```

True

Évolution de la file :

q

Files (utilisation)

```
q = Queue()
q.is_emtpy()
```

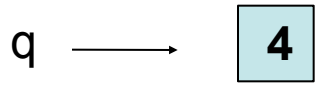
Évolution de la file :

q ⟶

Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
```

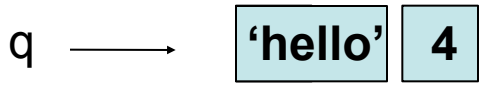
Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
```

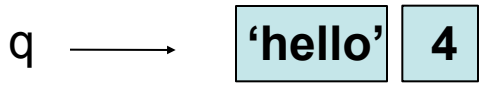
Évolution de la file :



Files (utilisation)

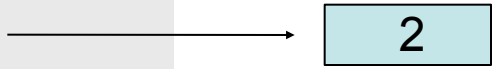
```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
```

Évolution de la file :



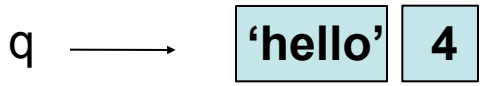
Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
```



2

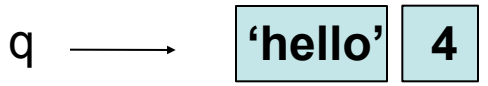
Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
```

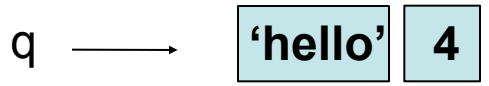
Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
```

Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
```

→ False

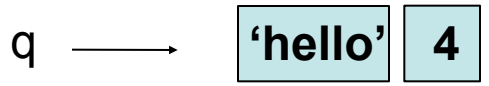
Évolution de la file :

q → **'hello'** **4**

Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
```

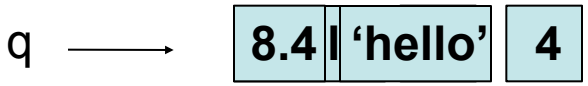
Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
```

Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
s.dequeue()
```

→ 4

Évolution de la file :

q → | **8.4** | **'hello'** |

Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
s.dequeue()
s.dequeue()
```

→ 'hello'

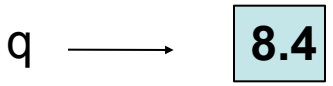
Évolution de la file :

q → **8.4**

Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
s.dequeue()
s.dequeue()
```

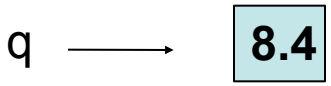
Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
s.dequeue()
s.dequeue()
s.printContent()
```

Évolution de la file :



Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
s.dequeue()
s.dequeue()
s.printContent()
```

→ 8.4

Évolution de la file :

q → **8.4**

Files (utilisation)

```
q = Queue()
q.is_emtpy()
q.enqueue(4)
q.enqueue('hello')
s.size()
s.is_empty()
s.enqueue(8.4)
s.dequeue()
s.dequeue()
s.printContent()
```

Évolution de la file :