

LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL KE-4

QUEUE PADA PYTHON



Disusun Oleh:

Nama : Oktario Mufti Yudha
NPM : 2320506044
Kelas : 04 (Empat)

Program Studi S1 Teknologi Informasi

Fakultas Teknik, Universitas Tidar

Genap 2023/2024

I. Tujuan Praktikum

Praktikum ini bertujuan untuk memperkenalkan mahasiswa pada konsep dasar Queue serta memberikan pemahaman praktis dalam mengimplementasikannya menggunakan bahasa pemrograman Python. Praktikum ini bertujuan untuk memberikan mahasiswa pemahaman yang kuat tentang bagaimana Queue beroperasi, termasuk konsep enqueue (menambahkan elemen), dequeue (menghapus elemen), dan peek (melihat elemen paling awal tanpa menghapusnya).

II. Dasar Teori

struktur data queue adalah konsep FIFO (First In, First Out), yang berarti elemen pertama yang dimasukkan ke dalam tumpukan adalah elemen pertama yang akan dikeluarkan. Struktur data ini mirip dengan antrian pada rumah sakit, dimana pasien yang pertama mendaftar adalah pasien yang akan dipanggil pertama kali.

Operasi dasar yang biasa digunakan adalah:

1. Enqueue (Menambahkan elemen): Operasi untuk menambahkan elemen ke dalam antrian. Elemen baru akan ditambahkan di ujung rear/tail dari antrian.
2. Dequeue (Menghapus elemen): Operasi untuk menghapus elemen dari antrian. Elemen yang dihapus adalah elemen yang berada di ujung front/head dari antrian.
3. Peek (Melihat elemen teratas): Operasi untuk melihat elemen teratas dalam antrian tanpa menghapusnya. Ini memungkinkan untuk memeriksa elemen yang berikutnya akan dihapus.

Tanda Tangan

III. Hasil dan Pembahasan

a. Queue dengan List

```
# Queue dengan List
queue = []

queue.append('a')
queue.append('i')
queue.append('u')
queue.append('e')
queue.append('o')

print('instal queue')
print(queue)

print('\nElement dequeue form queue:')
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

print('\nQueue after removing elements:')
print(queue)
```

instal queue
['a', 'i', 'u', 'e', 'o']

Element dequeue form queue:
a
i
u

Queue after removing elements:
['e', 'o']

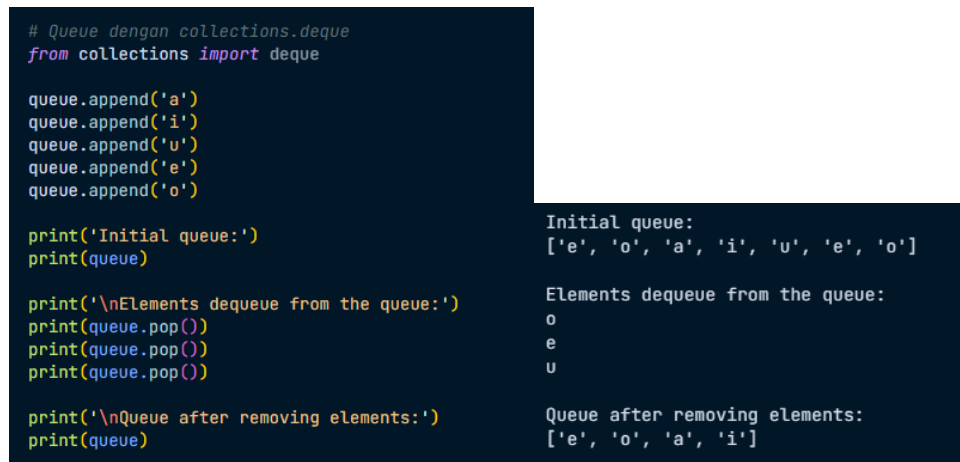
Gambar 3.1: Queue dengan List dan Outputnya

1. `queue = []`: Membuat sebuah list kosong yang akan digunakan untuk menyimpan elemen-elemen dalam antrian.
2. `queue.append('a')`: Huruf 'a' ditambahkan ke dalam antrian menggunakan metode `.append()`.
3. `queue.append('i')`: Huruf 'i' ditambahkan ke dalam antrian, menjadi elemen kedua dalam antrian.
4. `queue.append('u')`: Huruf 'u' ditambahkan ke dalam antrian, menjadi elemen ketiga dalam antrian.
5. `queue.append('e')`: Huruf 'e' ditambahkan ke dalam antrian, menjadi elemen keempat dalam antrian.
6. `queue.append('o')`: Huruf 'o' ditambahkan ke dalam antrian, menjadi elemen kelima dan terakhir dalam antrian.
7. `print('instal queue')`: Mencetak string 'instal queue'.
8. `print(queue)`: Mencetak isi dari antrian menggunakan fungsi `print()`.
9. `print("\nElement dequeue form queue:')`: Mencetak string '\nElement dequeue form queue:'.
10. `print(queue.pop(0))`: Menghapus dan mencetak elemen pertama dari antrian menggunakan metode `.pop(0)`. Huruf 'a' dihapus dari antrian dan dicetak.
11. `print(queue.pop(0))`: Menghapus dan mencetak elemen kedua dari antrian. Huruf 'i' dihapus dari antrian dan dicetak.

Tanda Tangan

12. `print(queue.pop(0))`: Menghapus dan mencetak elemen ketiga dari antrian. Huruf 'u' dihapus dari antrian dan dicetak.
13. `print("\nQueue after removing elements:')`: Mencetak string '\nQueue after removing elements:'.
14. `print(queue)`: Mencetak isi dari antrian setelah tiga elemen pertamanya dihapus.

b. Queue dengan `collections.deque`



```
# Queue dengan collections.deque
from collections import deque

queue.append('a')
queue.append('i')
queue.append('u')
queue.append('e')
queue.append('o')

print('Initial queue:')
print(queue)

print('\nElements dequeue from the queue:')
print(queue.pop())
print(queue.pop())
print(queue.pop())

print('\nQueue after removing elements:')
print(queue)
```

Initial queue:
['e', 'o', 'a', 'i', 'u', 'e', 'o']

Elements dequeue from the queue:
o
e
u

Queue after removing elements:
['e', 'o', 'a', 'i']

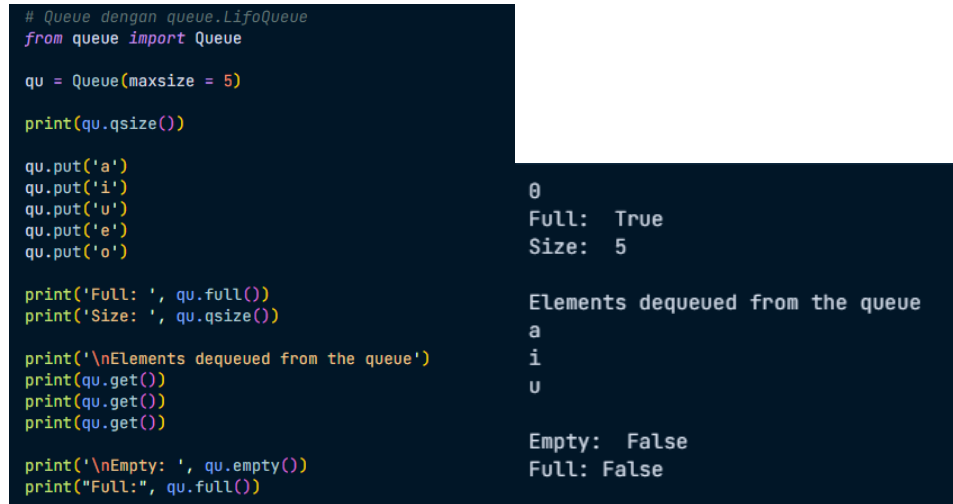
Gambar 3.2: Queue dengan collections.deque dan Outputnya

1. `from collections import deque` : Mengimpor kelas deque dari modul collections.
2. `queue.append('a')`, `queue.append('i')`, `queue.append('u')`, `queue.append('e')`, `queue.append('o')`: Huruf 'a, i, u, e, o' ditambahkan ke dalam antrian menggunakan metode `.append()`.
3. `print('Initial queue:')`: Mencetak string 'Initial queue:'.
4. `print(queue)`: Mencetak isi dari antrian menggunakan fungsi `print()`.
5. `print('\nElements dequeue from the queue:')`: Mencetak string '\nElements dequeue from the queue:'.
6. `print(queue.pop())`, `print(queue.pop())`, `print(queue.pop())`: Menghapus dan mencetak elemen-elemen dari antrian menggunakan metode `.pop()`.

Tanda Tangan

7. `print("\nQueue after removing elements:')`: Mencetak string `\nQueue after removing elements:'`.
8. `print(queue)`: Mencetak isi dari antrian setelah tiga elemen terakhirnya dihapus.

c. Queue dengan `queue.LifoQueue`



```
# Queue dengan queue.LifoQueue
from queue import Queue

qu = Queue(maxsize = 5)

print(qu.qsize())

qu.put('a')
qu.put('i')
qu.put('u')
qu.put('e')
qu.put('o')

print('Full: ', qu.full())
print('Size: ', qu.qsize())

print('\nElements dequeued from the queue')
print(qu.get())
print(qu.get())
print(qu.get())

print('\nEmpty: ', qu.empty())
print("Full:", qu.full())
```

```
0
Full:  True
Size:  5

Elements dequeued from the queue
a
i
u

Empty:  False
Full: False
```

Gambar 3.3: Queue dengan `queue.LifoQueue` dan output

1. `from queue import Queue`: Mengimpor kelas `Queue` dari modul `queue`.
2. `qu = Queue(maxsize = 5)`: Membuat objek antrian `qu` dengan ukuran maksimum (`maxsize`) sebesar 5.
3. `print(qu.qsize())`: Mencetak jumlah elemen saat ini dalam antrian menggunakan metode `.qsize()`.
4. `qu.put('a')`, `qu.put('i')`, `qu.put('u')`, `qu.put('e')`, `qu.put('o')`: Menambahkan huruf vokal ke dalam antrian menggunakan metode `.put()`.
5. `print('Full: ', qu.full())`: Mencetak apakah antrian sudah penuh menggunakan metode `.full()`, yang akan mengembalikan `True` jika antrian sudah penuh.
6. `print('Size: ', qu.qsize())`: Mencetak jumlah elemen saat ini dalam antrian menggunakan metode `.qsize()`.
7. `print("\nElements dequeued from the queue')`: Mencetak string `\nElements dequeued from the queue'`.

Tanda Tangan

8. `print(qu.get()), print(qu.get()), print(qu.get())`: Menghapus dan mencetak elemen-elemen dari antrian menggunakan metode `.get()`.
9. `print("\nEmpty: ', qu.empty())`: Mencetak apakah antrian kosong menggunakan metode `.empty()`, yang akan mengembalikan `True` jika antrian kosong.
10. `print("Full:", qu.full())`: Mencetak apakah antrian penuh menggunakan metode `.full()`.

d. Queue dengan `queue.Queue`



```
# Queue dengan queue.Queue
from queue import Queue

qu = Queue(maxsize = 5)

print(qu.qsize())

qu.put('a')
qu.put('i')
qu.put('u')
qu.put('e')
qu.put('o')

print('Full: ', qu.full())
print('Size: ', qu.qsize())

print('\nElements dequeued from the queue')
print(qu.get())
print(qu.get())
print(qu.get())

print('\nEmpty: ', qu.empty())
print("Full:", qu.full())
```

```
0
Full: True
Size: 5

Elements dequeued from the queue
a
i
u

Empty: False
Full: False
```

Gambr 3.4: Queue dengan queue.Queue

1. `from queue import Queue`: Mengimpor kelas `Queue` dari modul `queue`.
2. `qu = Queue(maxsize = 5)`: Baris ini membuat objek antrian `qu` dengan ukuran maksimum (`maxsize`) sebesar 5.
3. `print(qu.qsize())`: Mencetak jumlah elemen saat ini dalam antrian menggunakan metode `.qsize()`.
4. `qu.put('a'), qu.put('i'), qu.put('u'), qu.put('e'), qu.put('o')`: Menambahkan huruf vokal ke dalam antrian menggunakan metode `.put()`.
5. `print('Full: ', qu.full())`: Mencetak apakah antrian sudah penuh menggunakan metode `.full()`, yang akan mengembalikan `True` jika antrian sudah penuh.
6. `print('Size: ', qu.qsize())`: Mencetak jumlah elemen saat ini dalam antrian menggunakan metode `.qsize()`.

Tanda Tangan

7. `print("\nElements dequeued from the queue")`: Mencetak string `\nElements dequeued from the queue`.
8. `print(qu.get()), print(qu.get()), print(qu.get())`: Menghapus dan mencetak elemen-elemen dari antrian menggunakan metode `.get()`.
9. `print("\nEmpty: ", qu.empty())`: Mencetak apakah antrian kosong menggunakan metode `.empty()`.
10. `print("Full:", qu.full())`: Mencetak apakah antrian penuh menggunakan metode `.full()`.

e. Queue dengan Singel Linked List

```
# Queue dengan Singel Linked List
class Queue:
    def __init__(self):
        self.queue = list()

    def addtoqu(self, dataval):
        if dataval not in self.queue:
            self.queue.insert(0, dataval)
            return True
        return False

    def removefromqu(self):
        if len(self.queue) > 0:
            return self.queue.pop()
        return ("No elements in Queue!")

    def size(self):
        return len(self.queue)

TheQueue = Queue()
TheQueue.addtoqu("Jan")
TheQueue.addtoqu("Feb")
TheQueue.addtoqu("March")
TheQueue.addtoqu("April")
print(TheQueue.size())
print(TheQueue.removefromqu())
print(TheQueue.removefromqu())
print(TheQueue.size())
```

4
Jan
Feb
2

Gambr 3.5: Queue dengan Singel Linked List

1. `class Queue::` Mendefinisikan sebuah kelas Python bernama Queue.
2. `def __init__(self)::` Metode konstruktor kelas Queue, yang akan dipanggil ketika objek dari kelas ini dibuat.
3. `self.queue = list()`: Membuat list kosong yang akan digunakan untuk menyimpan elemen-elemen antrian.
4. `def addtoqu(self, dataval)::` Mendefinisikan metode `addtoqu()` untuk menambahkan elemen baru ke dalam antrian.
5. `if dataval not in self.queue::` Memeriksa apakah `dataval` belum ada dalam antrian.

Tanda Tangan

6. `self.queue.insert(0, dataval)`: Jika `dataval` belum ada dalam antrian, metode ini akan menambahkannya ke posisi awal antrian menggunakan metode `.insert()`.
7. `return True`: Mengembalikan `True` setelah menambahkan `dataval` ke dalam antrian.
8. `return False`: Mengembalikan `False` jika `dataval` sudah ada dalam antrian.
9. `def removefromqu(self)::` Mendefinisikan metode `removefromqu()` untuk menghapus elemen dari antrian.
10. `if len(self.queue) > 0::` Memeriksa apakah antrian tidak kosong.
11. `return self.queue.pop()`: Menghapus dan mengembalikan elemen terakhir dari antrian menggunakan metode `.pop()`.
12. `return ("No elements in Queue!")`: Mengembalikan pesan "No elements in Queue!" jika antrian kosong.
13. `def size(self)::` Mendefinisikan metode `size()` untuk mengembalikan jumlah elemen dalam antrian.
14. `return len(self.queue)`: Mengembalikan panjang (jumlah elemen) dari antrian menggunakan fungsi `len()`.
15. `TheQueue = Queue()`: Membuat sebuah objek `TheQueue` dari kelas `Queue`.
16. `TheQueue.addtoqu("Jan"), TheQueue.addtoqu("Feb"), TheQueue.addtoqu("March"), TheQueue.addtoqu("April")`: Menambahkan beberapa elemen ke dalam antrian menggunakan metode `addtoqu()`.
17. `print(TheQueue.size())`: Mencetak jumlah elemen dalam antrian menggunakan metode `size()`.
18. `print(TheQueue.removefromqu()), print(TheQueue.removefromqu())`: Menghapus dan mencetak elemen-elemen dari antrian menggunakan metode `removefromqu()`.
19. `print(TheQueue.size())`: Mencetak jumlah elemen dalam antrian setelah beberapa elemen dihapus.

Tanda Tangan

IV. Kesimpulan

Praktikum stack membantu kita memahami konsep dasar Queue dalam pemrograman. Kita belajar tentang cara menambahkan dan menghapus elemen pada Queue, serta bagaimana tumpukan bekerja menggunakan aturan FIFO (First In, First Out). Kita dapat melihat bagaimana Queue digunakan dalam pemrograman sehari-hari. Praktikum ini memberikan pemahaman yang kuat tentang struktur data ini dan bagaimana kita bisa menggunakan mereka dalam memecahkan masalah komputasi.

Tanda Tangan
