

# **LAPORAN PRAKTIKUM STRUKTUR DATA**

**MODUL KE-04**

**QUEUE DALAM PYTHON**



**Disusun Oleh:**

**Nama** : Restu Wibisono  
**NPM** : 2340506061  
**Kelas** : 03 (Tiga)

**Program Studi S1 Teknologi Informasi**

**Fakultas Teknik, Universitas Tidar**

**Genap 2023/2024**

## **I. Tujuan Praktikum**

Adapun tujuan praktikum ini sebagai berikut :

1. Mahasiswa mampu menerapkan konsep queue pada bahas pemrograman python.

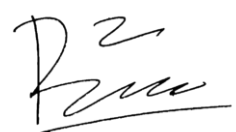
## **II. Dasar Teori**

Queue adalah salah satu struktur data yang umum digunakan dalam pemrograman. Konsep queue mengacu pada urutan yang dimasukkan dan dihapus dari struktur data. Prinsip dasar queue adalah FIFO (First In First Out), yang berarti elemen pertama masuk akan menjadi yang pertama keluar. Queue memiliki 2 ujung yaitu “ekor”(rear) adalah elemen yang baru saja ditambahkan, dan “kepala” (head) adalah elemen yang dihapus.

Queue memiliki beberapa operasi utama sebagai berikut:

1. Enqueue: Menambahkan elemen ke dalam queue. Operasi ini dilakukan pada rear queue. Saat queue sudah penuh maka bisa disebut dalam keadaan “Oerflow”. Waktu dalam pengoperasian queue adalah konstan ( $O(1)$ ) yang berarti tidak bergantung pada jumlah elemen di dalam queue.
2. Dequeue: Menghapus elemen dari queue. Operasi ini dilakukan pada Front queue yang sesuai dengan FIFO. Saat queue kosong, maka akan disebut sebagai ‘Underflow’. Waktu yang diperlukan juga konstan ( $O(1)$ ).
3. Front: Mengakses elemen yang berada di depan tanpa menghapusnya. Operasi ini bisa untuk melihat pada elemen pertama yang masuk ke dalam queue.
4. Rear: Mengakses elemen terakhir yang dimasukkan ke dalam queue. Operasi ini memungkinkan untuk melihat elemen yang terakhir dimasukkan atau ditambahkan.

Tanda Tangan



### III. Hasil dan Pembahasan

#### 1. Queue dengan List

```
# Python program to demonstrate queue implementation using list
queue = []

# append() function to enqueue element in the queue
queue.append('a')
queue.append('i')
queue.append('u')
queue.append('e')
queue.append('o')

print('instal queue')
print(queue)

# pop(0) function to dequeue element form queue in FIFO order
print('\nElement dequeue form queue:')
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

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

# Uncommenting print(queue.pop(0)) will raise and IndexError as the queue is now empty

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

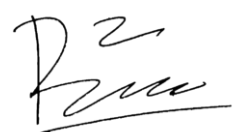
Element dequeue form queue:
a
i
u

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

(Gambar 3.1)

- Membuat list kosong 'queue = []' yang akan berfungsi sebagai antrian.
- Metode 'append()' berfungsi sebagai menambahkan elemen dalam antrian. Elemen-elemen berisikan 'a', 'i', 'u', 'e', 'o'.
- Program akan mencetak string 'instal queue' menggunakan perintah 'print()' untuk menandai bahwa antrian telah diinstal.
- Lalu mencetak antrian. Outputnya adalah list berisi elemen yang telah dimasukkan sebelumnya.
- 'print(queue.pop(0))' berfungsi untuk menghapus dan mengambil elemen dari antrian sesuai FIFO, dan elemen yang diambil adalah tiga elemen pertama.

Tanda Tangan



- Kemudian akan mencetak pesan ‘Queue after removing elements’ yang mana menandai bahwa setelah beberapa elemen telah dihapus.
- ‘print(queue)’ akan mencetak beberapa isi elemen setelah dihapus.

## 2. Queue dengan collections.deque

```
# Python program to demonstrate queue implementation using collections.deque
from collections import deque

# append() function to enqueue element in the queue
queue.append('a')
queue.append('i')
queue.append('u')
queue.append('e')
queue.append('o')

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

# pop() function to dequeue element from queue in LIFO order
print('\nElements dequeue from the queue:')
print(queue.pop())
print(queue.pop())
print(queue.pop())

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

# Uncommenting print(queue.pop()) will raise an IndexError as the queue is now empty
```

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

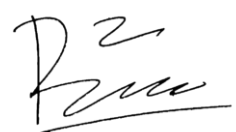
Elements dequeue from the queue:
o
e
u

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

(Gambar 3.2)

- Mengimport kelas ‘deque’ dari modul ‘collections’. ‘deque’ ini struktur data yang mirip dengan list akan tetapi lebih dioptimalkan untuk pengoperasian penambahan dan penghapusan di kedua ujungnya.
- Metode ‘append()’ berfungsi sebagai menambahkan elemen dalam antrian. Elemen-elemen berisikan ‘a’, ‘i’, ‘u’, ‘e’, ‘o’.
- Program akan mencetak string ‘instal queue’ menggunakan perintah ‘print()’ untuk menandai bahwa antrian telah diinstal.

Tanda Tangan



- Lalu mencetak antrian. Outputnya adalah 'deque(['a', 'i', 'u', 'e', 'o'])'.
- Selanjutnya program akan mencetak 'Elemen dequeue form the queue:' untuk menandai bahwa elemennya akan diambil dari antrian.
- Metode 'pop()' akan menghapus dan mengambil elemen dari antrian dalam antrian LIFO (Last In Frist Out), dan 3 elemen terakhir akan dicetak.
- Program akan mencetak 'Queue after removing elements:' untuk menandai bahwa beberapa elemen yang dicetak telah dihapus beberapa.
- Terakhir mencetak isi antrian setelah beberapa dari elemen dihapus. Jika tiga elemen telah dihapus maka akan tersisa dua 'a' dan 'i'.

### 3. Queue dengan queue.LifoQueue

```
# Python program to demonstrate implementation of queue using queue module
from queue import Queue

# Initializing a queue
qu = Queue(maxsize = 5)

# qsize() show the number of elements in the queue
print(qu.qsize())

# put() function to enqueue element in the queue
qu.put('a')
qu.put('i')
qu.put('u')
qu.put('e')
qu.put('o')

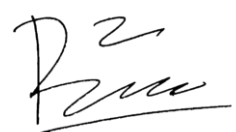
print('Full: ', qu.full())
print('Size: ', qu.qsize())

# get() function to dequeue element from queue
print('\nElements dequeued from the queue')
print(qu.get())
print(qu.get())
print(qu.get())

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

(Gambar 3.3.1)

Tanda Tangan



```

0
Full: True
Size: 5

Elements dequeued from the queue
a
i
u

Empty: False
Full: False

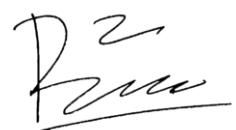
```

(Gambar 3.3.2)

- Mengimpor kelas 'Queue' dari modul 'queue'. Kelas ini berfungsi sebagai membuat objek antrian.
- Menginisialisasi sebuah object menjadi 'qu' dengan ukuran maksimum sebanyak 5 elemen.
- Mencetak jumlah elemen antrian. Pada tahap instalasi saat antrian masih kosong maka hasilnya adalah 0.
- Metode 'put()' berfungsi untuk memasukkan elemen ke antrian.
- Selanjutnya program akan menjalankan fungsi 'print('Full: ' + qu.full())' yang akan mencetak antrian penuh ataupun tidak, antrian memiliki nilai maksimal 5 dan diisi dengan 5 elemen yang hasilnya adalah 'True'.
- Lalu mencetak dari 'qu.size()' yang akan mencetak jumlah elemen dalam antrian, yang telah dimasukkan 5 elemen maka hasilnya adalah 5.
- Program akan mencetak 'Elemen dequeue form the queue:' untuk menandai bahwa elemennya akan diambil dari antrian.
- Memakai metode 'get()' untuk menghapus dan mengambil elemen dari antrian dalam FIFO, dan 3 elemen yang pertamanya akan dicetak.
- Terakhir adalah mencetak apakah antrian kosong atau tidak, karena telah diambil 3 maka hasilnya adalah 'False'.

#### 4. Queue dengan Singel Linked List

Tanda Tangan



```

class Queue:
    def __init__(self):
        self.queue = list()

    def addtoqu(self, dataval):
# Insert method to add element
        if dataval not in self.queue:
            self.queue.insert(0, dataval)
            return True
        return False

# Pop method to remove element
    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())

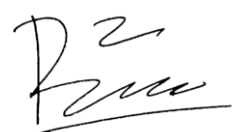
4
Jan
Feb
2

```

(Gambar 3.5)

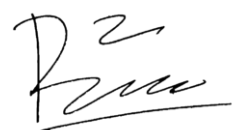
- Mendefinisikan kelas bernama 'Queue'.
- Menggunakan konstruktor '.\_\_init\_\_' yang akan dipanggil saat 'Queue' dibuat, yang akan menghasilkan atribut 'queue' sebagai list kosong.
- 'addtoqu' berfungsi untuk menambahkan elemen ke ke dalam antrian dengan parameter 'dataval' yang nilainya akan dimasukkan ke dalam antrian.
- Jika belum dalam antrian maka akan dimasukkan ke dalam indeks 0 memakai metode 'insert()'
- Jika dataval sudah dalam antrian, maka akan mengembalikan 'False'.

Tanda Tangan



- `'frmoveformqu(self)'` akan menghapus elemen dari antrian, jika panjang antrian lebih dari 0 maka elemen terakhir akan dihapus dari antrian menggunakan `'pop()'`.
- Selanjutnya `'def size(self)'` akan mengembalikan jumlah elemen dalam antrian.
- Lalu membuat object `'TheQueue'` dari kelas `'queue'`.
- Metode `'addtoqu()'` akan menambahkan beberapa elemen ke dalam antrian.
- `'print(TheQueue.size())'` akan mencetak jumlah elemen dalam antrian setelah elemen ditambahkan.

Tanda Tangan

A handwritten signature in black ink, consisting of a stylized 'P' followed by a series of loops and a horizontal line at the bottom.



#### IV. Kesimpulan

Dari praktikum pemrograman Python di ini dapat disimpulkan beberapa hal penting:

1. Queue sebagai Struktur Data Penting, karena dalam pengembangan perangkat lunak mengikuti prinsip FIFO dan konsep ini memiliki aplikasi yang sangat luas.
2. Operasi Dasar pada Queue yaitu enqueue (penambahan elemen), dequeue (penghapusan elemen), front (pengambilan elemen dari bagian depan antrian) dan rear (pengambilan elemen dari bagian belakang antrian). Serta dalam setiap operasi queue memiliki waktu konstan ( $O(1)$ ).
3. Implementasi dalam Python bisa menggunakan berbagai cara termasuk menggunakan list, koleksi 'deque' atau modul 'queue' yang menyediakan kelas 'Queue'.
4. Pemahaman Konsep Dasar dalam memahami queue dan operasi-operasinya. Dengan pemahaman yang baik bisa mendapatkan solusi terbaik dalam penyelesaian masalah dari setiap masalah yang dihadapi.

Dengan begitu akan memberikan pemahaman yang lebih baik dalam pemahaman konsep dan penggunaan queue dalam bahasa pemrograman python, serta akan membuat lebih siap saat menghadapi masalah yang ada dalam pengembangan perangkat lunak yang membutuhkan struktur data antrian.

Tanda Tangan

