Circular Queue

Federico Matthew Pratama - 233405001

Pertama buatlah sebuah Class bernama `CircularQueue`

```
class CircularQueue:
```

Setelah itu buatlah fungsi ketika program dijalankan akan menjalankan variable tersebut

```
def __init__(self, size):
  self.size = size
  self.queue = [None] * size
  self.front = self.rear = - 1
```

Setelah itu buatlah fungsi untuk pengecekan `is_empty` untuk cek apakah kosong atau `is_full` untuk cek apakah penuh

```
def is_empty(self):
  return self.front == - 1

def is_full(self):
  return (self.rear + 1) % self.size == self.front
```

Setelah itu buatlah fungsi untuk menambahkan data yaitu `Enqueue`

```
def enqueue(self, data):
  if self.is_full():
    print("Queue Penuh!")
    return

    if self.front == - 1:
      self.front = self.rear = 0
    else:
      self.rear = (self.rear + 1) % self.size
    self.queue[self.rear] = data
```

Penjelasan :

- Pertama cek apakah Queue masih full atau tidak
- Karena pada semula index di letakkan di luar list array, maka start front ke index 0 atau list pertama
- Ketika sudah masuk kedalam list, maka tambahkan data, geser ke kanan, lalu simpan datanya

Selanjutnya buatlah fungsi untuk menghapus data yaitu `Dequeue`

```
def dequeue(self):
  if self.is_empty():
    print("Queue Kosong!")
    return None

  removed = self.queue[self.front]

  if self.front == self.rear:
    self.front = self.rear = - 1
  else:
    self.front = (self.front + 1) % self.size
  return removed
```

Penjelasan :

- Pertama cek apakah Queue masih kosong atau tidak
- Variable `removed` ini digunakan untuk mengambil nilai pertama yang masuk (menggunakan metode FIFO atau First In First Out)
- Setelah itu ketika posisi front dan rear terletak di index yang sama, maka mundurin si index front (biar si front selalu dibelakang)
- Dilanjutkan dengan penggeseran index front
- Return removed ini agar mengirim nilai yang pertama masuk tadi ke variable `removed`

Setelah itu pembuatan Display

```
def display(self):
  result = []
  i = self.front
  while True:
    result.append(self.queue[i])
    if i == self.rear:
      break
    i = (i + 1) % self.size
  print(f"Queue : {result}")
```

Penjelasan :

- Pertama buat variable list bernama `result`
- Setelah itu ambil nilai `self.front` untuk posisi index hapus
- Lakukan looping dengan while, lalu masukkan data dari index `self.front` ke dalam list var `result` yang telah dibuat sebelumnya
- Ketika index `self.front` telah menyentuh ke index `self.rear` tandanya data telah penuh dan tidak bisa mengisi, maka keluar dari loop
- Terakhir index nya akan digeser
- Print Hasil

Full Code

```
class CircularQueue:
    def __init__(self, size):
        self.size = size
        self.queue = [None] * size
        self.front = self.rear = - 1

    def is_empty(self):
        return self.front == - 1

    def is_full(self):
        return (self.rear + 1) % self.size == self.front

    def enqueue(self, data):
        if self.is_full():
            print("Queue Penuh!")
            return

        if self.front == - 1:
            self.front = self.rear = 0
        else:
            self.rear = (self.rear + 1) % self.size
        self.queue[self.rear] = data

    def dequeue(self):
        if self.is_empty():
            print("Queue Kosong!")
            return None

        removed = self.queue[self.front]

        if self.front == self.rear:
            self.front = self.rear = - 1
        else:
            self.front = (self.front + 1) % self.size
        return removed

    def display(self):
        result = []
        i = self.front
        while True:
            result.append(self.queue[i])
            if i == self.rear:
                break
            i = (i + 1) % self.size
        print(f"Queue : {result}")
```